Kuvailevaa vaativuusteoriaa neuroverkoille

Veeti Ahvonen, Damian Heiman

Tampereen yliopisto

2023

Sisältö

Sisältö

Mitä on kuvaileva vaativuusteoria?

- Logiikassa on määritelty selitys- ja selitettävyysongelmat, joissa halutaan löytää joko pienin mahdollinen ekvivalentti kaava tai selitys sille, miksi kaava luokiteltiin tietyllä tavalla. Mutta entä jos halutaan selityksiä ratkaisuille, jotka koskevat muita systeemejä, kuten esimerkiksi neuroverkkoja?
- Kuvailevassa vaativuusteoriassa pyritään löytämään sellaisia logiikan kieliä, jotka vastaavat ilmaisuvoimaltaan täydellisesti tutkittavaa systeemiä. Tällöin kyseisen systeemin selitys- ja selitettävyysongelmat palautuvat kyseisen logiikan selitys- ja selitettävyysongelmiksi.
- Esimerkiksi: Jos löydetään logiikan kieli, joka vastaa täysin neuroverkkoja, niin kyseisiä neuroverkkoja voidaan minimoida vastaavin tavoin kuin kyseisen logiikan kaavoja. Neuroverkot ovat suunnattuja graafeja, jotka laskevat reaaliluvuilla; löytääksemme vastaavan logiikan kielen, pyrimme kääntämään neuroverkkojen sisäisen laskennan piireiksi eli suunnatuiksi syklittömiksi graafeiksi, jotka laskevat Boolen lukuarvoilla 0 ja 1 logiikan kaavojen tavoin.

Motivaatio

- Neuroverkot liittyvät tekoälyyn ja koneoppimiseen ja ovat erittäin ajankohtainen tutkimusaihe. Neuroverkossa ei toteuteta jotain ennalta tunnettua algoritmia tietokoneella, vaan tietokone saadaan oppimaan ratkaisuja itse. Tämä tapahtuu tutkimalla neuroverkon suorittamia luokitteluja ja hienosäätämällä sen parametreja, kunnes se osaa luokitella tarpeeksi hyvin. Neuroverkko voidaan opettaa esimerkiksi tunnistamaan käsin kirjoitettuja lukuja tai tunnistamaan kasvoja kuvista.
- Koska neuroverkko käytännössä oppii algoritmin itse, edes sen suunnittelija ei välttämättä tiedä, millä perusteella neuroverkko luokittelee syötteitä. Näin ollen neuroverkkojen selitettävyys on ajankohtainen tutkimusongelma. Kuvaileva vaativuusteoria antaa tähän työkaluja.

- Esimerkiksi EU:n yleinen tietosuoja-asetus edellyttää, että ihmisillä on oikeus tietää, millä tavalla heitä koskevia tietoja käytetään. Jos ihmiseltä evätään vakuutushakemus, ja hylkäämispäätöksen tekee neuroverkko, niin ihmisellä on oikeus tietää, minkä tiedon perusteella neuroverkko teki hylkäämispäätöksen. Neuroverkon tekemät luokittelut pitäisi siis pystyä selittämään.

Tästä tutkimuksesta jotain tietoa

- Tämä seminaari perustuu tutkimusartikkeliin "Descriptive complexity for neural networks via Boolean networks".
- Artikkelin tarkoitus on avata ovea neuroverkkojen selitettävyydelle löytämällä yleistä neuroverkkoluokkaa vastaava logiikan kieli. Avainasemassa on neuroverkkojen kääntäminen Boolen verkoiksi, missä käytetään reaalilukujen sijasta vain lukuja 0 ja 1. Kaikki neuroverkon sisäinen laskenta palautetaan binääriseen muotoon ja tulkitaan logiikan Boolean network logic (BNL) kielellä.
- Tuloksina saadaan, että neuroverkko pystytään tulkitsemaan logiikan BNL kaavoilla ja päinvastoin. Erityisesti tuloksista seuraa, että neuroverkkojen sisäiset epälineaariset ja joskus monimutkaiset "aktivointifunktiot" pystytään kääntämään hyvin yksinkertaiseen muotoon, joskin tämä kasvattaa neuroverkon kokoa.

Seminaarin sisältö

- Seminaari on kaksiosainen, eli kattaa sekä tiistain että torstain luennot.
- Aloitamme esittelemällä logiikan BNL_0 sekä sen laajennuksen BNL. Näistä voi tulla yksinkertaisia tehtäviä harjoitustehtäviin ja tenttiin. Voidaan esimerkiksi pyytää kirjoittamaan ohjelma, joka toimii tehtävänannossa tarkennetulla tavalla.
- Tämän jälkeen esitellään neuroverkot.
- Lopuksi esittelemme tutkimusartikkelin tulokset pääpiirteittäin.

Sisältö

BNL_0-ohjelma

Olkoon $\mathcal{T} = \{X, Y, Z\}$ järjestetty joukko ($X < Y < Z$), jossa on kolme Boolen muuttujaa (tämä tarkoittaa, että ne saavat arvoja 0 ja 1). Jokaiseen muuttujaan liitetään logiikan kaava, jossa käytetään jonon $\mathcal{T}$ muuttujia. Jokainen muuttuja saa alkuarvon (yläindeksi 0), joka on 0 tai 1. Jokainen muuttuja laskee itselleen uuden totuusarvon sijoittamalla muuttujien edelliset totuusarvot omaan kaavaansa. Tätä voidaan jatkaa mielivaltaisen kauan.

$$\begin{array}{l|llll} X :- Y \wedge Z, & X^0 = 0, & X^1 = 0, & X^2 = 1, & X^3 = 1, \quad \ldots \\ Y :- \neg X, & Y^0 = 0, & Y^1 = 1, & Y^2 = 1, & Y^3 = 0, \quad \ldots \\ Z :- X \vee Z. & Z^0 = 1, & Z^1 = 1, & Z^2 = 1, & Z^3 = 1, \quad \ldots \end{array}$$

Vasemmalla olevaa listaa, jossa on muuttujia ja niihin liitettyjä logiikan kaavoja, kutsutaan **logiikan BNL_0 ohjelmaksi** (Boolean network logic, suom. Boolen verkko -logiikka).

Syöte ja tuloste

Koska ohjelman muuttujat on järjestetty, niin muuttujien alkuarvot voidaan nähdä bittijonona $X^0 Y^0 Z^0 = 001$; tätä kutsutaan ohjelman **syötteeksi**. Ohjelman muuttujista valitaan myös tulostemuuttujat; valitaan tulostemuuttujiksi esimerkiksi muuttujat X ja Z. Ohjelman **tuloste** on niin ikään bittijono $X^t Z^t$, joka muodostuu tulostemuuttujien totuusarvoista jollakin ajanhetkellä $t \in \mathbb{N}$. Jos esimerkiksi määritellään, että ohjelma tulostaa ajanhetkellä $t = 2$, niin sen tuloste on $X^2 Z^2 = 11$.

$$
\begin{array}{l|llll}
X :- Y \wedge Z, & X^0 = 0, & X^1 = 0, & X^2 = 1, & X^3 = 1, \quad \ldots \\
Y :- \neg X, & Y^0 = 0, & Y^1 = 1, & Y^2 = 1, & Y^3 = 0, \quad \ldots \\
Z :- X \vee Z. & Z^0 = 1, & Z^1 = 1, & Z^2 = 1, & Z^3 = 1, \quad \ldots
\end{array}
$$

Ohjelman syöte on merkitty sinisellä ja tuloste punaisella.

Jos ohjelmalle annetaan eri syöte, se saattaa tuottaa eri tulosteen. Esimerkiksi syötteellä 010 sama ohjelma antaa tulosteeksi 00.

$$\begin{array}{l|llll} X :- Y \wedge Z, & X^0 = 0, & X^1 = 0, & X^2 = 0, & X^3 = 0, \quad \ldots \\ Y :- \neg X, & Y^0 = 1, & Y^1 = 1, & Y^2 = 1, & Y^3 = 1, \quad \ldots \\ Z :- X \vee Z. & Z^0 = 0, & Z^1 = 0, & Z^2 = 0, & Z^3 = 0, \quad \ldots \end{array}$$

Ohjelma voidaan siis nähdä laskentakoneena, joka kuvaa bittijonoja bittijonoille. Sillä pystyy toteuttamaan esimerkiksi kaksiarvoisia funktioita $\{0,1\}^k \to \{0,1\}^\ell$, missä $\ell \leq k$. Yllä oleva esimerkki toteuttaa erään funktion $\{0,1\}^3 \to \{0,1\}^2$.

Muuttujat

Olkoon $\mathrm{VAR} = \{ V_i \mid i \in \mathbb{N} \}$ äärettömästi numeroituva joukko **muuttujasymboleita** (tai yksinkertaisesti muuttujia) ja olkoon $<^{\mathrm{VAR}}$ muuttujien järjestys. Tyypillisesti käytämme symboleita $X, Y, Z, \ldots$ tai $X_1, X_2, \ldots$ viittaamaan joukon VAR alkioihin.

Jokainen joukko $\mathcal{T} \subseteq \mathrm{VAR}$ indusoi järjestyksen $<^{\mathcal{T}}$ joukon $\mathcal{T}$ alkioille. Toisin sanoen, jos $X, Y \in \mathcal{T}$ ja $X <^{\mathrm{VAR}} Y$, niin $X <^{\mathcal{T}} Y$. Saatamme yksinkertaisuuden vuoksi merkitä joukon $\mathcal{T}$ järjestystä ilman yläindeksiä $<$, jos joukko $\mathcal{T}$ on selvä kontekstista. Alaindeksejä käyttäessä oletamme, että muuttujat $X_1, \ldots, X_n$ ovat järjestyksessä $X_1 < X_2 < \cdots < X_n$.

BNL_0: Syntaksi

Määritelmä 1

Olkoon $\mathcal{T} = \{X_1, \ldots, X_n\} \subseteq \mathrm{VAR}$. **Logiikan** BNL_0 $\mathcal{T}$**-ohjelma** on kolmikko $(L, \mathcal{P}, t)$, missä L on lista

$$X_1 :- \psi_1,$$
$$\vdots$$
$$X_n :- \psi_n,$$

missä $\psi_1, \ldots, \psi_n$ ovat kielen $\psi ::= X_i \mid \neg\psi \mid \psi \wedge \psi$ määrittelemiä kaavoja ($X_i \in \mathcal{T}$), $\mathcal{P} \subseteq \mathcal{T}$ on joukko **tulostemuuttujia** ja $t \in \mathbb{N}$ on **tulostusajanhetki**.

Disjunktio, implikaatio ja ekvivalenssi voidaan määritellä lyhennysmerkintöinä tavalliseen tapaan.

BNL_0: Semantiikka

Olkoon $P = (L, \mathcal{P}, t)$ jokin logiikan BNL_0 $\mathcal{T}$-ohjelma ($\mathcal{P} \subseteq \mathcal{T} \subseteq \mathrm{VAR}, t \in \mathbb{N}$) ja olkoon $\pi\colon \mathcal{T} \to \{0,1\}$ funktio. Merkitään muuttujan $X \in \mathcal{T}$ totuusarvoa ajanhetkellä $t \in \mathbb{N}$ yläindeksillä: X^t. Se määritellään induktiolla seuraavasti:

- Ajanhetkellä 0 määritellään, että $X^0 = \pi(X)$.
- Oletetaan sitten, että muuttujien totuusarvot on määritelty ajanhetkellä $t \in \mathbb{N}$. Ajanhetkellä $t+1$ määritellään, että $X^{t+1} = 1$, mikäli muuttujaan X liittyvä kaava ψ on tosi, kun siinä esiintyvien muuttujien totuusarvot tulkitaan ajanhetkellä t. (Tässä ψ on siis se kaava, jolla $X :- \psi$ esiintyy ohjelmassa.)

Ohjelman P **syöte** on bittijono $X_1^0 \cdots X_n^0$, missä $\mathcal{T} = \{X_1, \ldots, X_n\}$.
Ohjelman P **tuloste** on bittijono $Y_1^t \cdots Y_\ell^t$, missä $\mathcal{P} = \{Y_1, \ldots, Y_\ell\}$.
Huomaa, että muuttujat on järjestetty indeksien mukaan.

Esimerkki: Leksikografinen järjestys

Esimerkki 1

Olkoon $\mathcal{T} = \{X_1, \ldots, X_n\}$. Tarkastellaan ohjelmaa

$$X_1 :- \left(\neg X_1 \wedge \bigwedge_{k=2}^{n} X_k\right) \vee \left(X_1 \wedge \bigvee_{k=2}^{n} \neg X_k\right),$$

$$X_2 :- \left(\neg X_2 \wedge \bigwedge_{k=3}^{n} X_k\right) \vee \left(X_2 \wedge \bigvee_{k=3}^{n} \neg X_k\right),$$

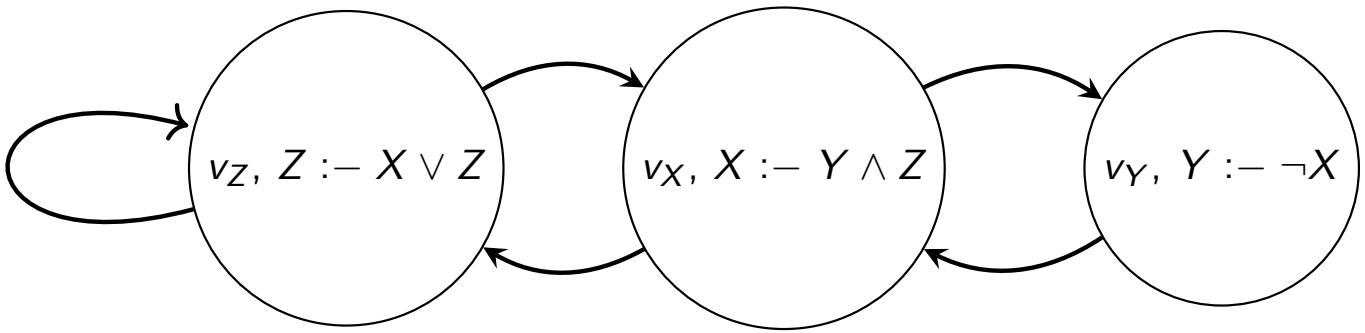
$$\vdots$$

$$X_n :- \neg X_n.$$

Ohjelman toimintaa voi kuvata yksinkertaisesti siten, että se käy läpi kaikki n:n pituiset bittijonot leksikografisessa järjestyksessä. Jos esimerkiksi $n = 3$ ja syöte 000, niin ohjelma käy läpi jonot järjestyksessä $000 \to 001 \to 010 \to 011 \to 100 \to 101 \to 110 \to 111 \to 000 \to \ldots$.

Boolen verkot

Nimensä mukaan Boolen verkko -logiikan ohjelma voidaan esittää (suunnattuna) verkkona. Ohjelman muuttujat X tulkitaan solmuina v_X. Solmusta v_X on särmä solmuun v_Y täsmälleen silloin, kun muuttuja X esiintyy muuttujaan Y liitetyssä kaavassa. Esimerkiksi edellisten diojen esimerkissä muuttuja X esiintyy muuttujan Y kaavassa $\neg X$, joten vastaavassa graafissa on särmä solmusta v_X solmuun v_Y. Voidaan ajatella, että solmut lähettävät totuusarvonsa toisilleen särmiä pitkin ja muuttavat tilaansa sijoittamalla vastaanottamansa viestit kaavaansa.



Kuva: BNL_0-ohjelman $X :- Y \wedge Z;\ Y :- \neg X;\ Z :- X \vee Z$ esitys suunnattuna graafina.

BNL-ohjelma

Logiikan BNL_0 ohjelmissa tulosteen pituus on aina korkeintaan syötteen pituus. Voi olla toteutuksen kannalta hyödyllistä tai jopa tarpeellista määritellä ohjelmaan **apumuuttujia**, ja näin päästään myös eroon syötteen ja tulosteen pituuksien riippuvuussuhteesta. Apumuuttujat eivät saa syötettä, vaan niillä on aina sama totuusarvo laskennan alussa. Muokataan edellistä ohjelmaa valitsemalla, että X ja Z ovat apumuuttujia.

$$\begin{array}{lll|lllll} X^0 = 1, & X :- Y \wedge Z, & & X^0 = 1, & X^1 = 0, & X^2 = 0, & \ldots \\ & Y :- \neg X, & & Y^0 = 1, & Y^1 = 0, & Y^2 = 1, & \ldots \\ Z^0 = 0, & Z :- X \vee Z. & & Z^0 = 0, & Z^1 = 1, & Z^2 = 1, & \ldots \end{array}$$

Nyt ohjelman syötteen koko on vain 1, sillä muuttujat X ja Z eivät saa syötettä. Ohjelma toteuttaa erään funktion $\{0,1\} \to \{0,1\}^2$, ja se antaa syötteellä 1 tulosteeksi 01. Ohjelmaa, jossa on kaksi listaa – yksi, joka kertoo apumuuttujien alkuarvot, ja toinen, joka kertoo muuttujia vastaavat logiikan kaavat – kutsutaan **logiikan** BNL **ohjelmaksi**.

BNL: syntaksi

Määritelmä 2

Olkoon $\mathcal{T} = \{X_1, \ldots, X_n\} \subseteq \mathrm{VAR}$. **Logiikan** BNL $\mathcal{T}$**-ohjelma** on kolmikko $(L, \mathcal{P}, t)$, missä L on lista

$$\begin{aligned} Y_1^0 &= b_1, & X_1 &:- \psi_1, \\ &\vdots & &\vdots \\ Y_m^0 &= b_m, & X_m &:- \psi_m \\ & & &\vdots \\ & & X_n &:- \psi_n, \end{aligned}$$

missä $\{Y_1, \ldots, Y_m\} \subseteq \mathcal{T}$, $b_1, \ldots, b_m \in \{0, 1\}$, $\psi_1, \ldots, \psi_n$ ovat kielen $\psi ::= X_i \mid \neg\psi \mid \psi \wedge \psi$ määrittelemiä kaavoja ($X_i \in \mathcal{T}$), $\mathcal{P} \subseteq \mathcal{T}$ on joukko **tulostemuuttujia** ja t on **tulostusajanhetki**.

Saman muuttujan ehdot kannattaa käytännössä asettaa vierekkäin.

BNL: semantiikka

Olkoon $P = (L, \mathcal{P}, t)$ jokin logiikan BNL $\mathcal{T}$-ohjelma ($\mathcal{P} \subseteq \mathcal{T}, t \in \mathbb{N}$) ja olkoon $\pi\colon \mathcal{T} \to \{0,1\}$ funktio. Merkitään muuttujan $X \in \mathcal{T}$ totuusarvoa ajanhetkellä $t \in \mathbb{N}$ yläindeksillä: X^t. Se määritellään induktiolla seuraavasti:

- Ajanhetkellä 0 määritellään, että $X^0 = b$, jos ohjelmassa esiintyy ehto $X^0 = b$ jollakin $b \in \{0,1\}$. Muussa tapauksessa $X^0 = \pi(X)$.
- Oletetaan sitten, että muuttujien totuusarvot on määritelty ajanhetkellä $t \in \mathbb{N}$. Muuttujan X totuusarvo ajanhetkellä $t+1$ määritellään täsmälleen samoin kuin logiikassa BNL_0.

Ohjelman P **syöte** on bittijono $X_1^0 \cdots X_n^0$, missä $X_1, \ldots, X_n$ ovat täsmälleen ne ohjeman P muuttujat, jotka eivät ole apumuuttujia.
Ohjelman P **tuloste** on bittijono $Y_1^t \cdots Y_\ell^t$, missä $\mathcal{P} = \{Y_1, \ldots, Y_\ell\}$.

Esimerkki: Liputus

Esimerkki 2

Olkoon $\mathcal{T} = \{F, X, Y, Z\}$. Tarkastellaan BNL-ohjelmaa

$$
\begin{aligned}
F^0 = 0, \quad & F :- F \vee \neg F, \\
& X :- (\neg F \wedge (Y \wedge Z)) \vee (F \wedge X), \\
& Y :- (\neg F \wedge \neg X) \vee (F \wedge Y), \\
& Z :- (\neg F \wedge (X \vee Z)) \vee (F \wedge Z).
\end{aligned}
$$

Ohjelma on muunnelma aiemmasta BNL_0-ohjelmasta; muuttujien X, Y ja Z silloiset säännöt on merkitty sinisellä, mutta nyt ne on ehdollistettu muuttujalla F, jota kutsutaan lipuksi (eng. flag). Oli ohjelman syöte mikä tahansa, niin $F^t = F^1$, $X^t = X^1$, $Y^t = Y^1$ ja $Z^t = Z^1$ kaikilla $t \geq 2$. Toisin sanoen ohjelma soveltaa BNL_0-ohjelman sääntöjä tasan kerran. BNL_0-ohjelma voi tuottaa saman tulosteen ajanhetkellä $t = 1$, mutta se ei kykene kierrosten laskemiseen BNL-ohjelman tavoin.

Sisältö

Mitä on koneoppiminen?

- Koneoppiminen (eng. machine learning) epämuodollisesti tarkoittaa ongelmien ratkaisemista siten, että kone “opetetaan” löytämään mahdollisimman hyvä algoritmi ongelman ratkaisuun sen sijaan että algoritmi olisi määritelty itse.
- Koneoppimisen "koneita"on useita ja näistä keskeisempiä tämän hetken tutkimuksessa ovat neuroverkot.

Esimerkki 3

Tarkastellaan konetta, jonka on tarkoitus tunnistaa, onko kuvassa kissa vai koira. Jos kone antaa väärän vastauksen, sille kerrotaan siitä, ja kone yrittää säätää algoritmin paremmaksi esim. tarkastelemalla kuvan otuksen korvien muotoa tarkemmin.

Neuroverkon johdanto

Neuroverkko (eng. neural network) on koneoppimiskone, jolla on verkkomainen rakenne ja jonka idea on simuloida aivoja. Neuroverkot ovat siis opetettavia laskentakoneita. Usein neuroverkot käyttävät laskemiseen reaalilukuja ja niillä on syöte sekä tuloste.

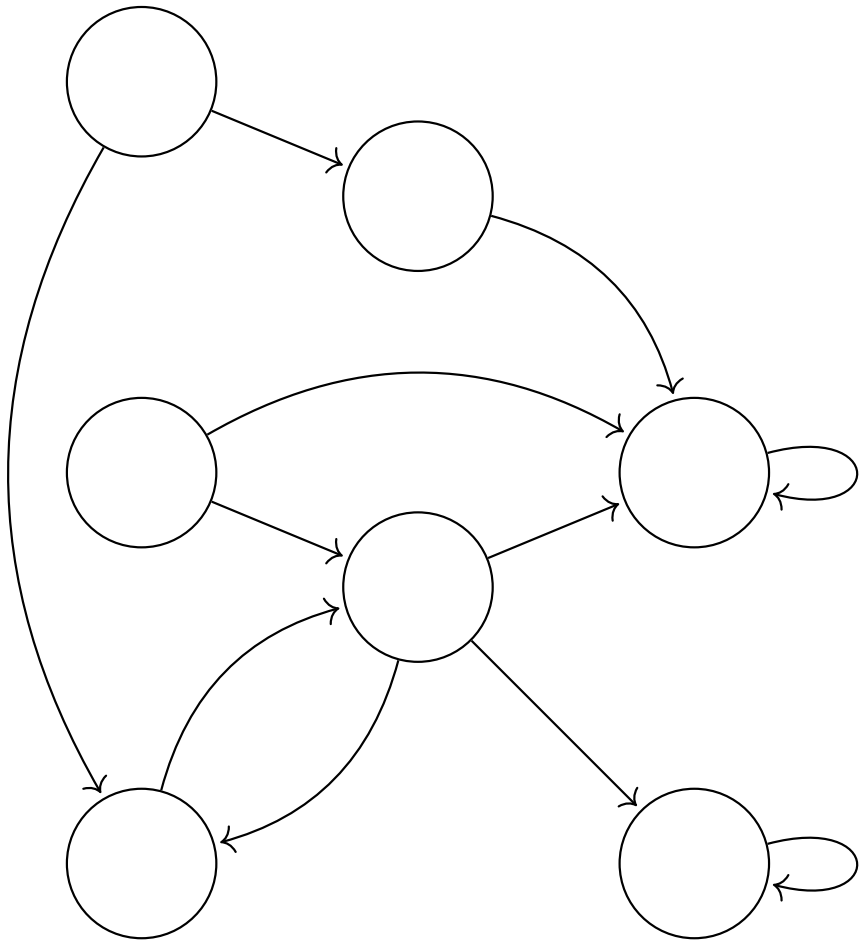
Katsotaan ensin johdatteleva pätkä 3Blue1Brownin videosta (suosittelemme katsomaan videon myös loppuun omalla ajalla). Kts. 3Blue1Brown video kohta 0.0 - 5.31

Kyseisessä videossa on kuvattu ns. “feedforward neural network”, jossa layerit vastaavat neuroverkon kierroksia ja verkon topologia on rajoitettu.

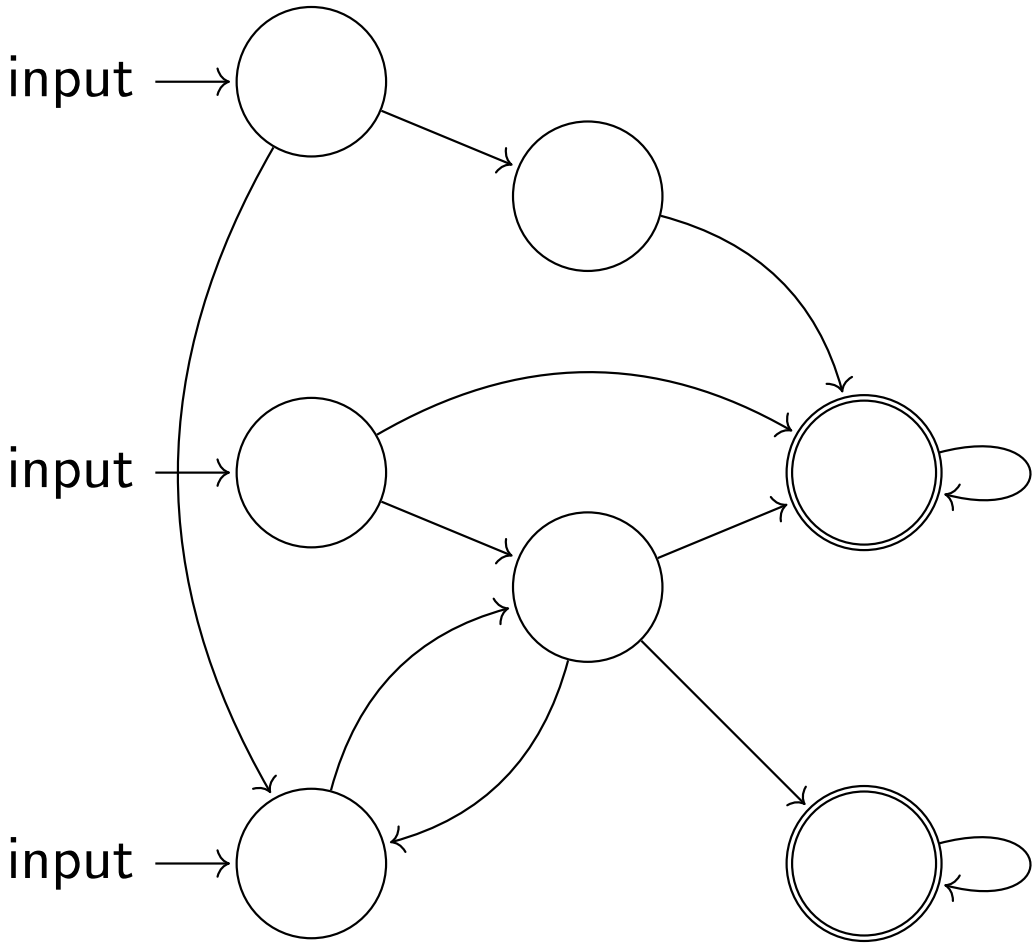
Tarkoituksena on esitellä asteen yleisempi neuroverkon määritelmä.

Aloitetaan ensin hyvin epämuodollisella neuroverkon kuvailulla siten, että konstruoimme esimerkki-neuroverkon vaihe vaiheelta. Sen jälkeen käymme neuroverkon toiminnan epämuodollisesti läpi ja mainitsemme neuroverkon opettamisesta.

Formaali määritelmä tulee myöhemmin. Tärkeintä on se, että ymmärtää pääpiirteittäin, miten neuroverkko toimii.



Neuroverkolla on suunnatun verkon rakenne (vähän kuin aivoilla). Kirjallisuudessa solmuja saatetaan nimittää **neuroneiksi** ja särmiä **synapseiksi**.



Neuroverkossa on usein nimetyt **syötesolmut** (eng. input nodes) ja **tulostesolmut**. Merkitään syötesolmuja piirtämällä niihin nuoli “input” ja tulostesolmuja tuplareunoilla. Kirjallisuudessa syötteenä ovat usein reaaliluvut.

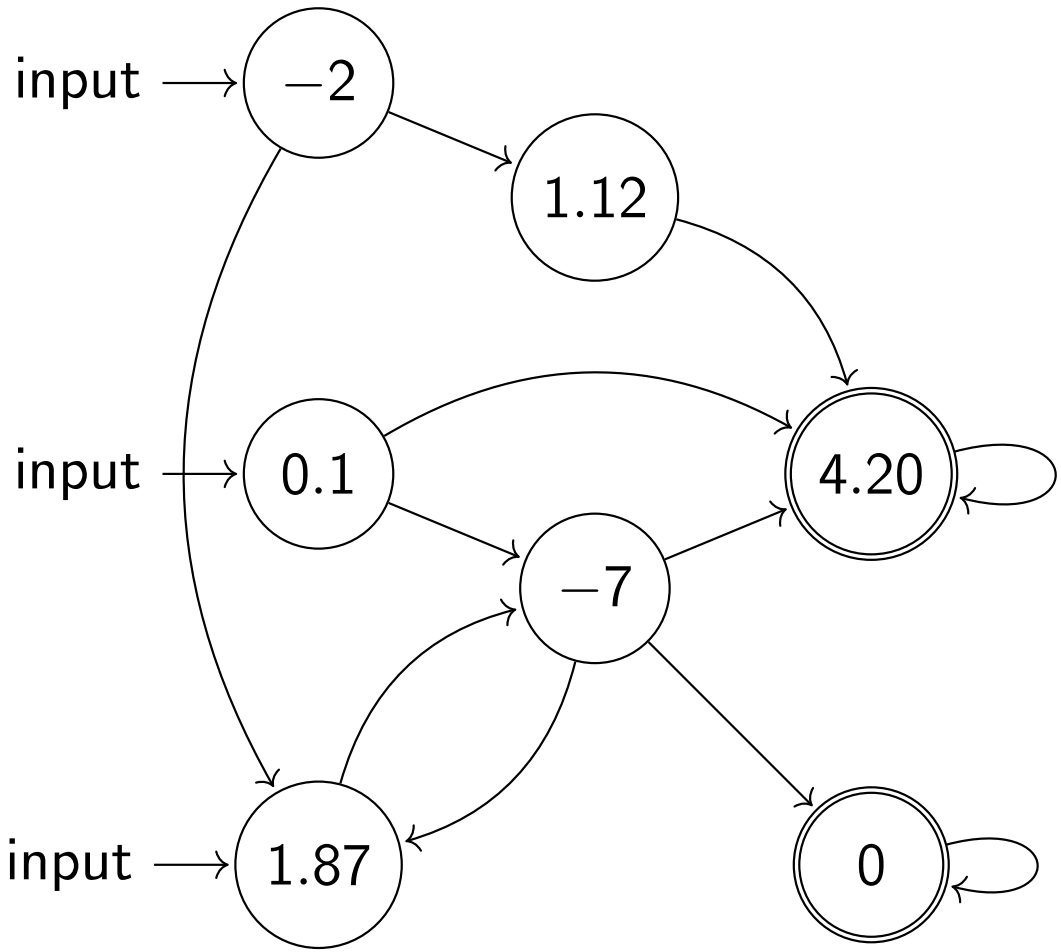
Neuroverkon toiminnan johdanto

Ennen kuin neuroverkon rakennetta tarkennetaan, kuvaillaan mitä sen olisi tarkoitus tehdä. Koska neuroverkkojen on tarkoitus simuloida aivoja, niin neuronien välinen viestiminen tulisi kuvata.

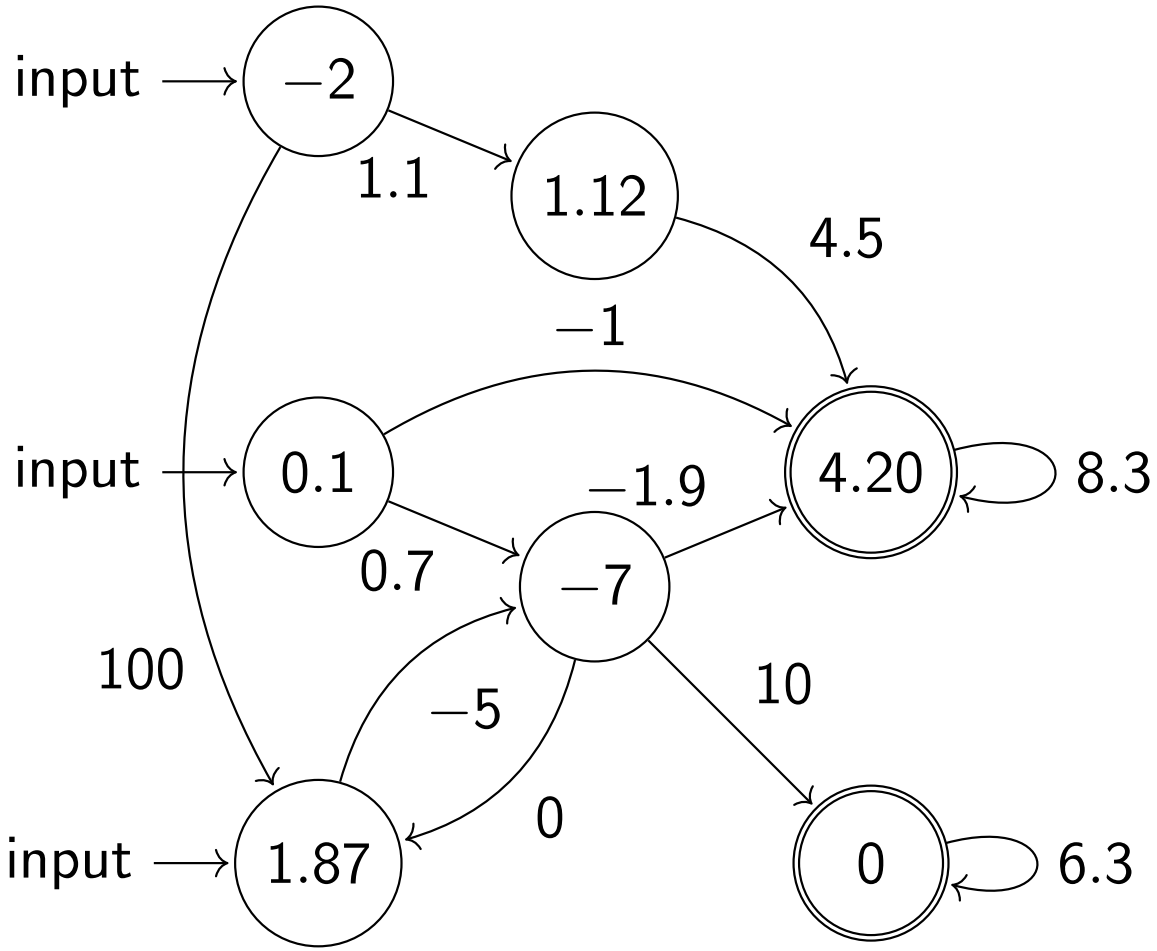
- Yksittäisellä neuronilla on **aktiivisuusarvo** (eng. activation value), joka on saatu syötteenä tai laskemalla. Usein kirjallisuudessa aktiivisuusarvo on reaaliluku. Intuitiivisesti voidaan ajatella, että mitä suurempi luku on sitä "vahvempi" on aktiivisuusarvo ja mitä pienempi sitä "heikompi" se on.
- Neuroverkko toimii kierroksittain: Alkuun syötesolmut saavat aktiivisuusarvonsa syötteenä "käyttäjältä" ja muut saavat ennalta määrätyn aktiivisuusarvon. Neuronit lähettävät aktiivisuusarvonsa särmiä pitkin (nuolien suuntaan) toisille neuroneille, joka kierros. Saaduista aktiivisuusarvoista lasketaan uusi aktiivisuusarvo. Tietyn ajan jälkeen saadaan sitten tuloste, joka määräytyy tulostesolmuista.

Se, milloin neuroverkko tulostaa, määritellään ulkopuolisella “kellolla”. Kello määrää **tulostekierroksen**, joka kertoo milloin tulosteen on oltava valmis. Voidaan esim. määritellä, että neuroverkko tulostaa tasan kolmen kierroksen jälkeen. Tämä on analogista BNL-ohjelmien kanssa, joissa valittiin kierros $t \in \mathbb{N}$, jolloin tuloste on valmis.

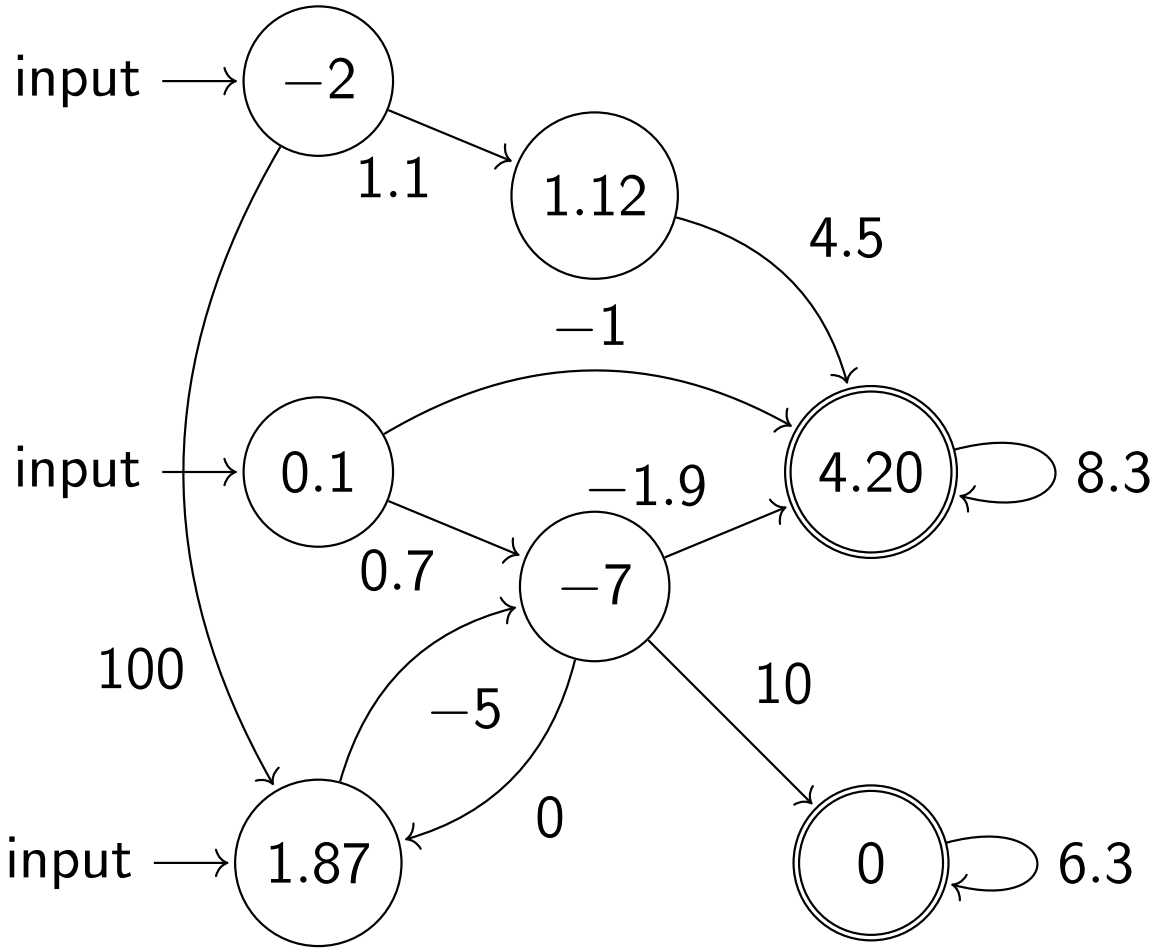
Jotta näitä aktiivisuusarvoja voitaisiin laskea, täytyy neuroverkon rakennetta tarkentaa. Täydennetään seuraavaksi aikaisemman esimerkin neuroverkkoa monimutkaisemmaksi yksi askel kerrallaan. Idea on, että neuroverkko laskee aktiivisuusarvoja reaaliluvuilla.



Jokainen solmu v on varustettu **korjaustermillä** b_v, joka on kirjallisuudessa usein reaaliluku.



Jokainen särmä e on varustettu **painolla** w_e, joka on usein kirjallisuudessa myös reaaliluku.



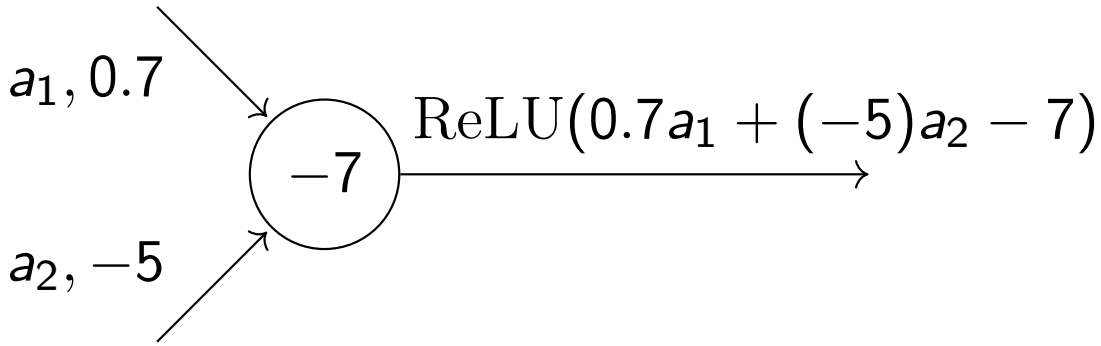
Loppuun vielä, jokainen solmu v on varustettu **aktivointifunktiolla** (eng. activation function) α_v, joka on usein reaalifunktio $\mathbb{R} \to \mathbb{R}$. Se voi olla eri joka solmussa, mutta käytännön sovelluksissa ne ovat usein samat. Tunnettuja aktivointifunktioita ovat esim. $\mathrm{ReLU}(x) = \max\{0, x\}$ ja sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$. Kts. Wikipedia-artikkeli.

Yksittäisen neuronin toiminta epämuodollisesti

Otetaan tarkasteluun edellisen dian neuroni, jossa on korjausterminä arvo -7, ja nimitetään sitä solmuksi s. Oletetaan, että jokaisella neuroverkon neuronilla v on jokin aktiivisuusarvo a_v ja että solmussa s on aktivointifunktio $\mathrm{ReLU}(x) = \max\{0, x\}$. Neuroniin tulee näin ollen 2 särmää, joista kullakin on oma painonsa. Uusi aktiivisuusarvo neuronille s lasketaan seuraavissa vaiheissa:

1. Neuronit, joista tulee särmä solmuun s, lähettävät oman aktiivisuusarvonsa neuroniin s.
2. Särmiä pitkin tulevat aktiivisuusarvot kerrotaan särmäkohtaisella painolla.
3. Nämä kerrotut aktiivisuusarvot summataan keskenään yhdessä korjaustermin kanssa.
4. Edellisen vaiheen summa syötetään aktivointifunktioon, joka laskee uuden aktiivisuusarvon.

Tämä on havainnollistettu seuraavassa diassa.



Kuvassa a_1 ja a_2 kuvastavat särmiä pitkin tulevia aktiivisuusarvoja toisista solmuista. Uusi aktiivisuusarvo saadaan näin ollen kaavalla $\mathrm{ReLU}(0.7a_1 + (-5)a_2 - 7)$. Jos esim. $a_1 = -2$ ja $a_2 = -0.9$, niin saadaan $\mathrm{ReLU}(0.7(-2) + (-5)(-0.9) - 7) = \mathrm{ReLU}(-1.4 + 4.5 - 7) = \mathrm{ReLU}(-3.9) = 0$.

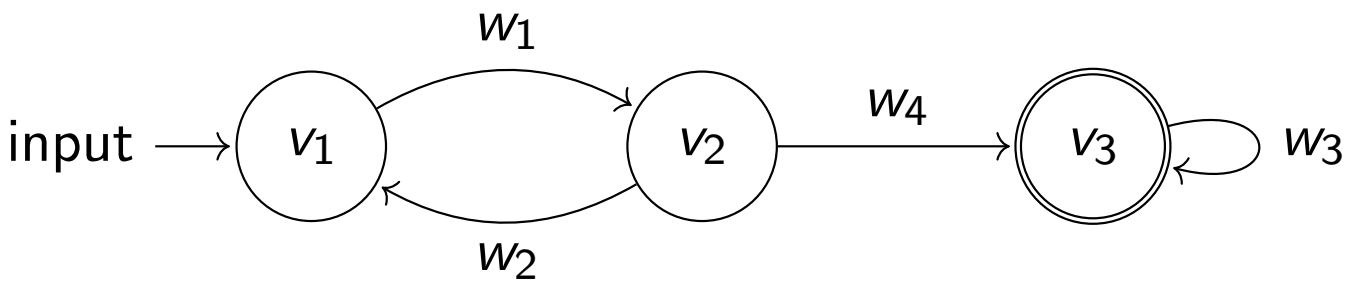
Äskeisen dian neuronin toiminta voidaan yleistää seuraavasti mille tahansa neuronille v. Oletetaan, että α_v on neuronin v aktivointifunktio, $a_1, \ldots, a_n$ ovat sen edeltäjien aktiivisuusarvot ja $w_1, \ldots, w_n$ ovat vastaavat painot kullekin edeltäjälle, ja että b_v on neuronin v korjaustermi. Tällöin neuronin v uusi aktiivisuusarvo voidaan laskea kaavalla

$$\alpha_v\left(\sum_{i=1}^{n} a_i w_i + b_v\right).$$

- Painojen voidaan ajatella kertovan, kuinka tärkeä särmää pitkin tuleva aktiivisuusarvo on. Mitä tärkeämpi se on, sitä suurempi on särmän paino.
- Painotettu summa edeltäjien aktiivisuusarvoista intuitiivisesti kertoo kuinka "aktiivisia" neuronin edeltäjät ovat.
- Korjaustermin voidaan ajatella kuvaavan kuinka herkästi painotettu summa edeltäjien aktiivisuusarvoista huomioidaan. Esim. jos korjaustermi on -7, niin painotetun summan on oltava vähintään 7, että se olisi huomion arvoinen.
- Aktivointifunktion voidaan intuitiivisesti ajatella kuvaavan kuinka aktiivinen neuronin pitää olla saadusta painotetusta summasta yhdessä korjaustermin kanssa. Yleensä mitä suurempi on aktivointifunktion syöte, sitä suurempi on aktivointifunktion antama arvo (eli yleensä aktivointifunktio on monotoninen).

Koko neuroverkon toiminta epämuodollisesti

Kuvassa on yksinkertainen neuroverkko. Neuroverkko toimii kierroksissa $k \in \mathbb{N}$.



- Jos $k = 0$, niin neuroni v_1 saa aktiivisuusarvonsa syötteenä $s \in \mathbb{R}$ "käyttäjältä", koska se on syötesolmu. Neuronien v_2 ja v_3 aktiivisuusarvot on ennalta määrätty ns. alkuarvoina eikä käyttäjä voi niihin vaikuttaa. Solmun v_2 alkuarvo voisi olla 0 ja solmun v_3 alkuarvo voisi olla -1.
- Jos $k = 1$, niin neuroni v_1 lähettää kierroksella 0 saadun syötteen s neuroniin v_2, neuroni v_2 lähettää alkuarvonsa neuroneihin v_1 ja v_3, ja neuroni v_3 lähettää alkuarvonsa itselleen. Sen jälkeen jokainen neuroni soveltaa aiemman dian mukaista kaavaa $\alpha_v\left(\sum_{i=1}^{n} a_i w_i + b_v\right)$ saatuihin aktiivisuusarvoihin. Näin jokainen neuroni saa uuden arvon.

Edellisen dian proseduuria voidaan jatkaa loputtomiin. Toisella kierroksella ($k = 2$) neuronit lähettävät kierroksella 1 laskemansa aktiivisuusarvot samaan tapaan särmiä pitkin neuroneille ja laskevat uuden aktiivisuusarvon samaan tapaan. Sama metodi toistuu myös seuraavilla kierroksilla.

Mikä sitten on neuroverkon tuloste? Tuloste voidaan määrätä ulkopuolisella "kellolla", joka kertoo millä kierroksella $t \in \mathbb{N}$ tulosteen on oltava valmis. Esim. kolmen kierroksen jälkeen tulosteen on oltava valmis.

Koko neuroverkon toiminta asteen muodollisemmin

Esitämme hiukan formaalimmin neuroverkon toiminnan. Aktiivisuusarvot lasketaan kierroksissa $k \in \mathbb{N}$ seuraavasti.

1. Jos $k = 0$, niin jokainen ei-syötesolmu saa jonkin alkuarvon. Olkoon V neuroverkon neuronien joukko ja I syötesolmujen joukko. Alkuarvot ei-syötesolmuille on ennalta määrätty **alustusfunktiolla** (eng. initializing function) $V \setminus I \to \mathbb{R}$. Syötesolmut saavat syötteen jonka antaa "käyttäjä" funktiona $I \to \mathbb{R}$.
2. Jos $k \geq 1$, niin jokaiselle neuronille v lasketaan uusi aktiivisuusarvo kierroksen $k-1$ neuronin v edeltäjien aktiivisuusarvoista $a_1, \ldots, a_n$ aiemman dian kaavan $\alpha_v\left(\sum_{i=1}^{n} a_i w_i + b_v\right)$ mukaan.

Neuroverkossa on määrätty ennalta **tulostekierros** $t \in \mathbb{N}$ ja **tulostesolmut** $T \subseteq V$ (missä V on neuroverkon solmujen joukko).

Neuroverkon formaali määritelmä

Määritelmä 3

Neuroverkko on järjestetty verkko $(V, E, <^V)$ (eli suunnattu verkko jossa solmut on järjestetty $<^V$ mukaan), jossa on lisäksi

- jokaisella särmällä $e \in E$ on **paino** $w_e \in \mathbb{R}$,
- jokaisella solmulla $v \in V$ on **korjaustermi** (eng. bias) $b_v \in \mathbb{R}$,
- jokaisella solmulla $v \in V$ on **aktivointifunktio** $\alpha_v \colon \mathbb{R} \to \mathbb{R}$,
- **alustusfunktio** $\pi \colon U \to \mathbb{R}$, missä $U \subseteq V$,
- **tulostekierros** $t \in \mathbb{N}$,
- **tulostesolmut** $T \subseteq V$.

Eli tarkalleen ottaen neuroverkko on jono, jossa nämä otetaan mukaan: $((V, E, <^V), (w_e)_{e \in E}, (b_v)_{v \in V}, (\alpha_v)_{v \in V}, \pi, t, T)$.
Joukkoa $V \setminus U$ sanotaan **syötesolmujen** (eng. input nodes) joukoksi.

Lyhyesti opettamisesta

Olemme puhuneet paljon neuroverkon toiminnasta, mutta miten niihin liittyy opettaminen?

Lyhyesti opettaminen neuroverkoissa liittyy painojen ja korjaustermien säätämiseen: Opettamatonta neuroverkkoa ajetaan jollain syötteellä ja katsotaan, kuinka se siitä suoriutuu. Jos tulos on epämiellyttävä, niin säädetään painoja ja korjaustermejä “paremmaksi”. Siihen, miten niitä säädetään, on kehitetty omia menetelmiä.

Sisältö

Motivaatio uudestaan

Kuvailevan vaativuusteorian tavoitteena olisi saada **täydellinen vastaavuus** neuroverkkojen ja BNL-logiikan välille eli ns. antaa **looginen karakterisointi** neuroverkoille. Puhumme myöhemmin ns. **kääntämisestä**. Kun looginen karakterisointi on saatu, niin se mahdollistaa logiikan selitettävyyden tutkimisen ja näin myös neuroverkkojen selitettävyyden tutkimisen logiikan kautta. Erityisen kiinnostava on yleinen selitysongelma.

Olemme neuroverkkojen kuvailevassa vaativuusteoriassa kiinnostuneita siitä minkä kokoiset BNL-ohjelmat vastaavat neuroverkkoja ja päinvastoin. Lisäksi olemme kiinnostuneita siitä, kuinka nopeasti neuroverkkoa vastaava BNL-ohjelma tuottaa tulosteen ja vastaavasti kuinka nopeasti BNL-ohjelmaa vastaava neuroverkko tuottaa tulosteen. Aika ja koko ovat siis kiinnostavia, ja pyrimme pitämään ne mahdollisimman "kesyinä" käännöksissä.

Kesy tässä kontekstissa tarkoittaa epämuodollisesti sitä, että jos olio x, jonka "koko" (tai "tulostuskierros") on m, käännetään vastaavaksi olioksi y, niin y:n "koko"' (tai "tulostuskierros") on $p(m)$, missä p on jokin polynomi. Käännösten koot tai ajat eivät siis "räjähdä" eksponentiaalisiksi.

Koko käsitteenä jätetään tässä esitelmässä epämääräiseksi neuroverkoille ja BNL-ohjelmille. Karkeasti BNL-ohjelmissa tämä tarkoittaa muuttujien lukumäärää ja sääntöjen pituuksien summaa. Neuroverkon "koko" koostuu taas useammasta osasta: solmuista, särmistä ja siitä, kuinka "vaativia" ovat aktivointifunktiot. Tätä ei ole aikaa selittää sen tarkemmin. Riittää ymmärtää intuitio.

Nopeasti huomataan yksi merkittävä ero ja **ongelma** BNL-logiikan ja neuroverkkojen välillä: neuroverkko käyttää laskemiseen reaalilukuja ja BNL-logiikka totuusarvoja 0 ja 1.

Ratkaisu: Approksimoidaan reaaleja **liukuluvuilla** (eng. floating-point). Tähän on hyvin perusteltu lähtökohta, nimittäin tietokoneet, joilla neuroverkkoja sovelletaan, käyttävät laskemiseen myös **liukulukujen aritmetiikkaa** (eng. floating-point arithmetic). Käydään liukulukuihin liittyviä käsitteitä *epämuodollisesti* läpi.

Jos x on jokin reaaliluku ja $m \in \mathbb{N}_+$ (positiiviset kokonaisluvut), niin merkitään $\mathrm{fl}(x, m)$ luvun x esitystä m mittaisena liukulukuna.

Esimerkki 4

Tunnetusti luvun π esitys on ääretön, mutta se voidaan esittää liukulukuna esim. $\mathrm{fl}(\pi, 1) = 3$, $\mathrm{fl}(\pi, 3) = 3.14$ ja $\mathrm{fl}(\pi, 11) = 3.1415926535$ ovat π:n liukulukuesityksiä. Mitä pidempi esitys on, sitä tarkempi se on.

Oletamme tästä eteenpäin neuroverkkojen käyttävän laskemiseen liukulukuja. Lisäksi oletamme aina, että liukulukujen esitykselle eli pituudelle on jokin yläraja. Tällaista neuroverkkoa nimitetään **käytännön neuroverkoksi**. Huomaa, että tarkalleen ottaen aiemmin määriteltyä reaaliluvuilla toimivaa neuroverkkoa voisi nimittää "ideaaliksi" neuroverkoksi. Käytännön sovelluksissa nämä esitykset ovat suhteellisen pieniä yksittäiselle luvulle (puhutaan kymmenien tai muutamien satojen pituisista merkkijonoista). Jos N on neuroverkko ja m liukulukujen esityksen yläraja neuroverkossa, merkitään käytännön neuroverkkoa N lyhyesti parina (N, m).

Esimerkki 5

Jos neuroverkon N liukulukujen yläraja on 3, niin luvun 1 esitys neuroverkossa on silloin 1.00.

Lisäksi, liukuluvut voidaan esittää monella tapaa bittijonojen avulla esim. IEEE 754 -standardi. Jos F on liukuluku, niin merkitään **liukuluvun bittiesitystä** $\mathrm{bit}(F)$.

Simuloinnista

Ennen kuin päälauseet voidaan muotoilla, niin määritellään *epämuodollisesti* simuloinnin käsitteet, jotka kertovat vastaavuudet BNL-ohjelmien ja neuroverkkojen välillä.

Määritelmä 4

Olkoon (N, m) käytännön neuroverkko ja P BNL-ohjelma. Oletetaan, että N:n syötesolmujen lkm. on n ja tulostesolmujen lkm. k. Sanomme, että P **simuloi** käytännön neuroverkkoa (N, m), jos seuraava ehto pätee.

Jokaisella n-mittaisella jonolla $(F_1, \ldots, F_n)$ liukuluja (joiden pituus kork. m) pätee: Jos N tuottaa syötteellä $(F_1, \ldots, F_n)$ k-mittaisen jonon liukulukuja $(O_1, \ldots, O_k)$ (joiden pituus myös kork. m), niin P tuottaa syötteellä $\mathrm{bit}(F_1) \cdots \mathrm{bit}(F_n)$ tulosteen $\mathrm{bit}(O_1) \cdots \mathrm{bit}(O_k)$.

Vastaavasti määritellään simulointi toiseen suuntaan.

Määritelmä 5

Olkoon (N, m) käytännön neuroverkko ja P BNL-ohjelma. Oletetaan, että P:n syötesolmujen lkm. on n ja tulostesolmujen lkm. k. Sanomme, että (N, m) **simuloi** ohjelmaa P, jos seuraava ehto pätee.

Jokaisella bittijonolla $b_1 \cdots b_n \in \{0,1\}^n$: Jos P tuottaa syötteellä $b_1 \cdots b_n$ bittijonon $o_1 \cdots o_k \in \{0,1\}^k$, niin (N, m) tuottaa syötteellä $(\mathrm{fl}(b_1, m), \ldots, \mathrm{fl}(b_n, m))$ tulosteen $(\mathrm{fl}(o_1, m), \ldots, \mathrm{fl}(o_k, m))$.

Päätuloksien muotoilu

Määritellään seuraavaksi tutkimustuloksien päälauseet *epämuodollisesti*.

Lause 1

Olkoon (N, m) käytännön neuroverkko. Tällöin on olemassa BNL*-ohjelma P, joka simuloi käytännön neuroverkkoa (N, m). Lisäksi, P:n koko on kesy suhteessa neuroverkon kokoon ja vastaavasti P:n tulostekierros on kesy suhteessa neuroverkon tulostekierrokseen.*

Lause 2

Olkoon P BNL*-ohjelma. Tällöin on olemassa käytännön neuroverkko (N, m), joka simuloi ohjelmaa P ja jonka jokainen aktivointifunktio on* ReLU*. Lisäksi, neuroverkon (N, m) koko on kesy suhteessa ohjelman kokoon ja neuroverkon tulostekierros on kesy suhteessa ohjelman tulostekierrokseen.*

Jälkimmäistä lausetta on mahdollisuus laajentaa muille aktivointifunktiolle, mutta sitä ei tässä esitelmässä käydä läpi.

Kahdesta edellisen dian lauseesta saadaan näin ollen suora seuraus.

Seuraus 1

Jokaista käytännön neuroverkkoa (N, m) *vastaa käytännön neuroverkko* (N^*, m)*, jonka jokainen aktivointifunktio on* ReLU*.*

Vastaavuus tarkoittaa tässä kontekstissa sitä, että neuroverkot "tuottavat samalla syötteellä saman tulosteen."

Todistusideat

Esitämme hyvin epämuodolliset ja karkeat päälauseiden todistusideat. Näitä täydennetään luennoilla, jos jää aikaa. Käydään ensin läpi Lauseen 1 todistuksen idea:

Jokaiselle neuroverkon neuronille v muodostetaan aliohjelma P_v, joka simuloi neuroverkon neuronin v sisäistä laskentaa. Tätä varten meidän tarvitsee "simuloida" kaksi asiaa:

1. liukulukujen aritmetiikka ja
2. aktivointifunktion toiminta.

Esim. yhteenlaskun simulointi tarkoittaa, että jos kahdesta liukuluvusta F_1 ja F_2 saadaan liukuluku F, niin vastaava BNL-ohjelma syötteellä $\mathrm{bit}(F_1)\mathrm{bit}(F_2)$ tuottaa tulosteen $\mathrm{bit}(F)$.

Kun jokaisen yksittäisen neuronin toimintaa kuvaava aliohjelma P_v on muotoiltu, näistä muodostetaan neuroverkkoa simuloiva ohjelma P.

Lauseen 2 todistusidea:

Ensin muodostamme annetusta BNL-ohjelmasta P vastaavan BNL-ohjelman P', jossa jokainen sääntö on yksinkertainen: sääntö saa sisältää korkeintaan yhden konnektiivin. Tämän jälkeen ohjelmasta P' muodostetaan neuroverkko. Neuroverkon rakenne vastaa suoraan P':n verkkoesitystä ja neuroverkkoon säädetään sopivat painot sekä korjaustermit, joilla ReLU osaa laskea oikean arvon.

Esim. tarkastellaan solmua, jossa on sääntö $X \wedge Y$. Tällöin vastaavaan neuroverkon neuroniin v tulee kaksi särmää. Näihin särmiin laitetaan painot 1 ja korjaustermiksi solmuun v asetetaan -1. Neuroverkolle annetaan alkuarvoiksi tai syötteeksi arvoja $0, 1$, jolloin neuroni v osaa simuloida sääntöä $X \wedge Y$. Perustelu, painotettu summa tulevista arvoista yhdessä korjaustermin kanssa on korkeintaan 1 ja tämä tapahtuu jos molemmat tulevat aktiivisuusarvot ovat 1, jolloin aktivointifunktio ReLU antaa arvoksi 1. Muilla tulevilla aktiivisuusarvoilla neuroni antaa uudeksi aktiivisuusarvoksi 0 eli niin kuin pitääkin.

Yhteenveto

- Esiteltiin logiikat BNL_0 ja BNL, sekä niiden verkkoesitykset.
- Esiteltiin neuroverkot ja verrattiin sitä logiikkojen verkkoesityksiin.
- Tehtiin ero ideaalin ja käytännön neuroverkkojen välille.
- Osoitettiin, että käytännön neuroverkoilta on kesy käännös BNL-ohjelmiksi ja päinvastoin.

Harjoituksiin ja tenttiin voi tulla helppoja tehtäviä BNL_0- ja BNL-logiikoista. Neuroverkoista ja tutkimustuloksista ei tule tehtäviä harjoituksiin tai tenttiin.

Huom! Ensi viikon torstain luento (28.9.2023) järjestetään salissa B1100. Tämän jälkeen torstain luennot siirtyvät Paavo Koli -saliin A2100. Tiistain luentosalit eivät muutu.