Myötäsyötteiset neuroverkot

Jimi Käyrä

15.12.2022

Matemaattiset tieteet

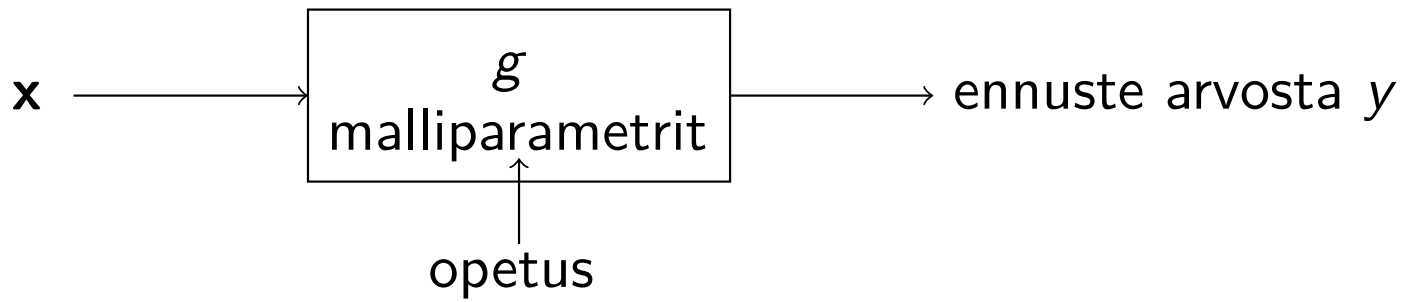
Ohjatun oppimisen malli

- Mallia opetetaan opetusdatajoukolla

$$L = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\} \subset X \times Y,$$

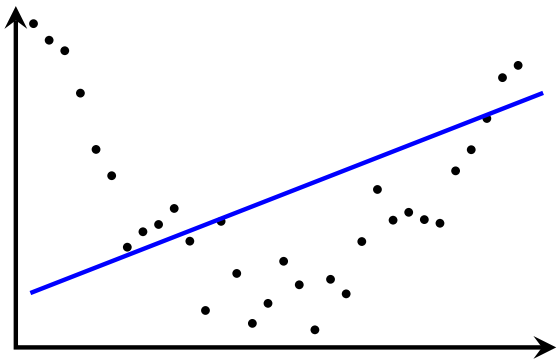
  jossa $x^{(i)} \in X$ on opetusesimerkin i piirrevektori ja $y^{(i)} \in Y$ vastaava kohdemuuttujan arvo.

- Tavoitteena on etsiä sellainen kuvaus $g : X \to Y$, että $g(x)$ vastaa mahdollisimman hyvin tuntematonta arvoa, kun $x \in X$ on opetusdataan kuulumaton piirrevektori.

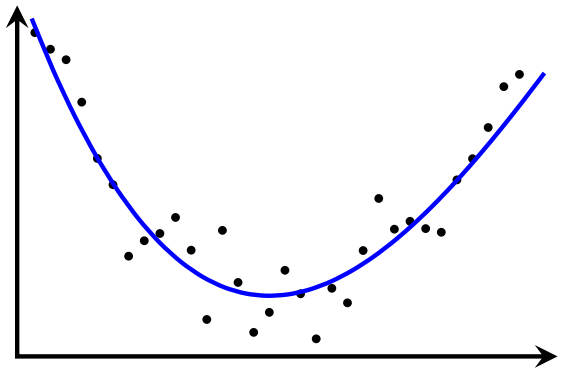


Opetuksen ongelmia

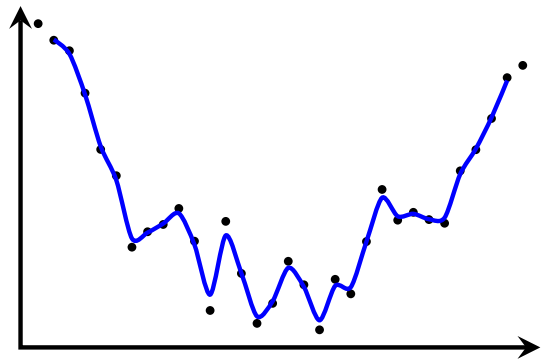
Alioppiminen



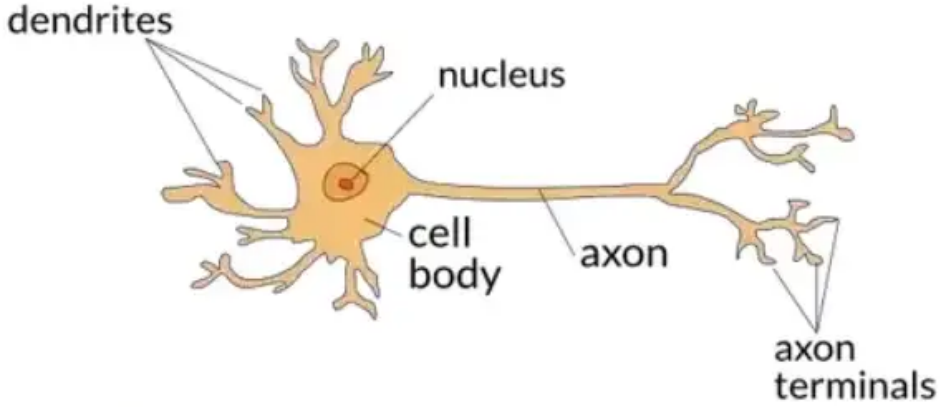
"kultainen keskitie"

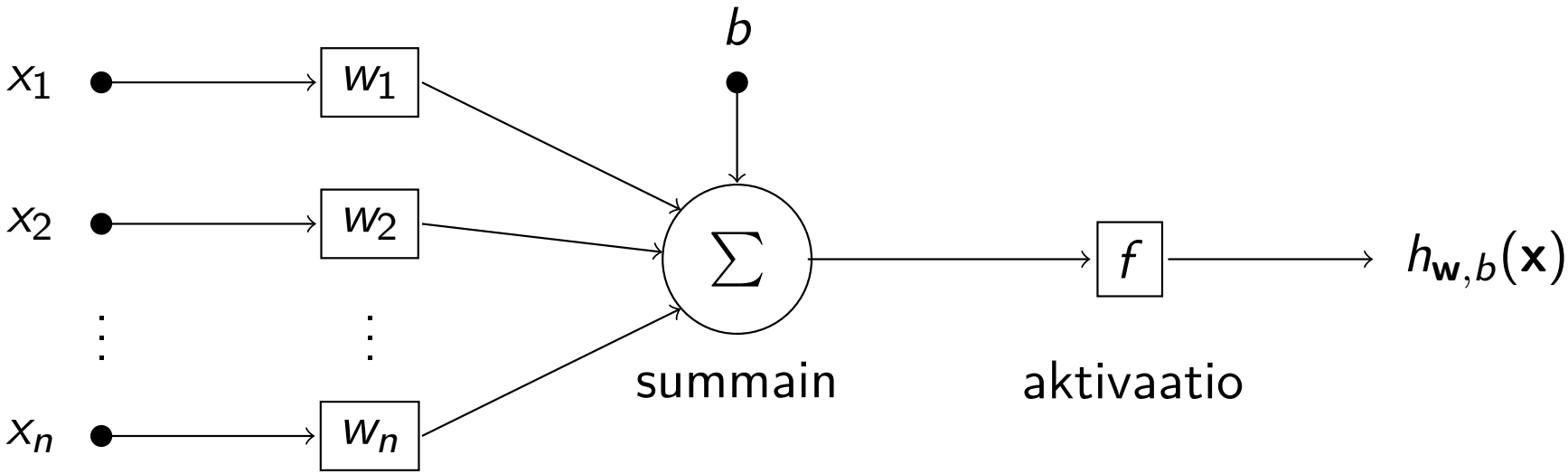


Ylioppiminen



Keinotekoinen neuroni



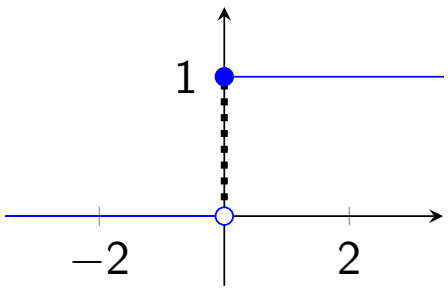


$$h_{\mathbf{w},b}(\mathbf{x}) = f\left(b + \sum_{i=1}^{n} w_i x_i\right) = f(\mathbf{w}^\top \mathbf{x} + b)$$

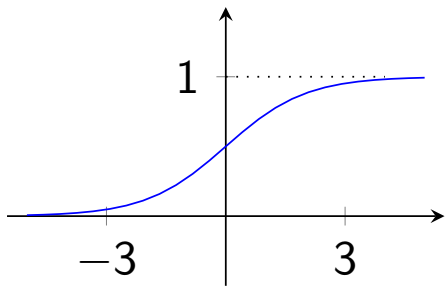
Aktivaatiofunktio

- Summaimen lähtö $\mathbf{w}^\top \mathbf{x} + b$ on syötteen $\mathbf{x}$ affiini muunnos
  $\Rightarrow$ aktivaatiofunktio f mahdollistaa epälineaarisuuden
- Kynnystys (vrt. biologinen neuroni)

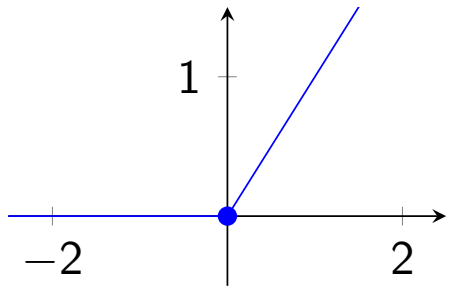
binäärinen askelfunktio



sigmoidi



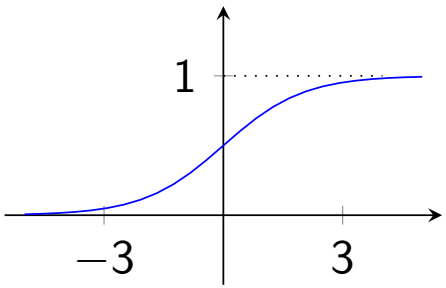
ReLU

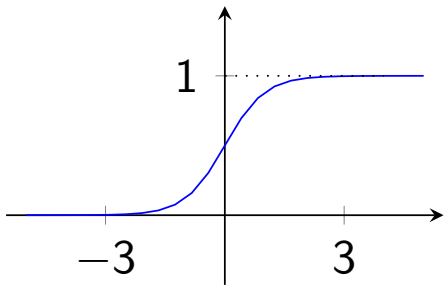


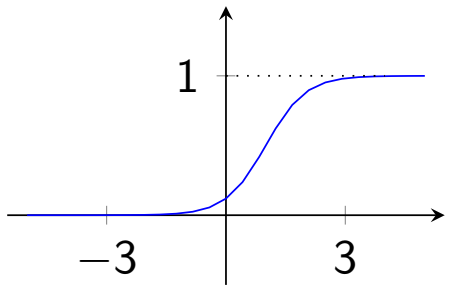
Bias

- Aktivaatioon vaadittavan kynnyksen säätö
- Voidaan mieltää vakiosyötteeksi

$$\boxed{y = f(x_1)} \qquad \boxed{y = f(w_1 x_1)} \qquad \boxed{y = f(w_1 x_1 + b)}$$







Neuroverkon arkkitehtuuri

Määritelmä

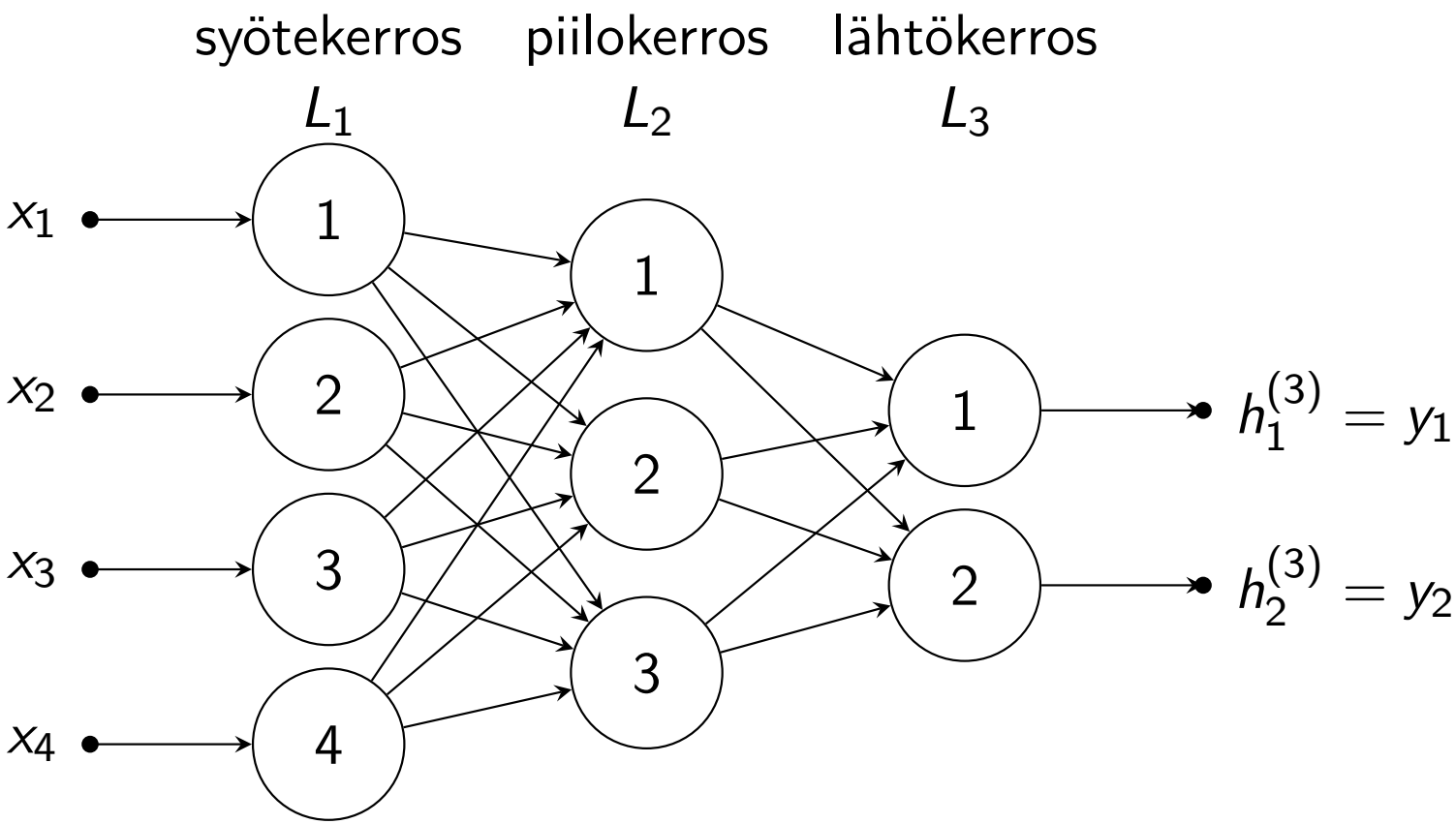
Neuroverkon arkkitehtuuri voidaan kuvata järjestettynä nelikkona (I, L, O, E), jossa I on syötepaikkojen joukko, L laskentayksikkösolmujen joukko, O lähtöpaikkojen joukko ja E joukko, joka koostuu painotetuista, suunnatuista linkeistä. Linkki on kolmikko (u, v, w), jossa $u \in I \cup L$, $v \in L \cup O$ ja $w \in \mathbb{R}$.

Määritelmä

Neuroverkko on myötäsyötteinen (*feed-forward*) täsmälleen silloin, kun se ei sisällä syklejä.

Neuroverkon arkkitehtuuri

- Merkitään
  - $w_{ij}^{(l)}$: kerroksen L_{l+1} neuronia i ja kerroksen L_l neuronia j yhdistävän linkin paino,
  - $b_i^{(l)}$: kerroksen L_{l+1} neuronin i bias-termi,
  - $h_i^{(l)}$: kerroksen L_l neuronin i lähtö.

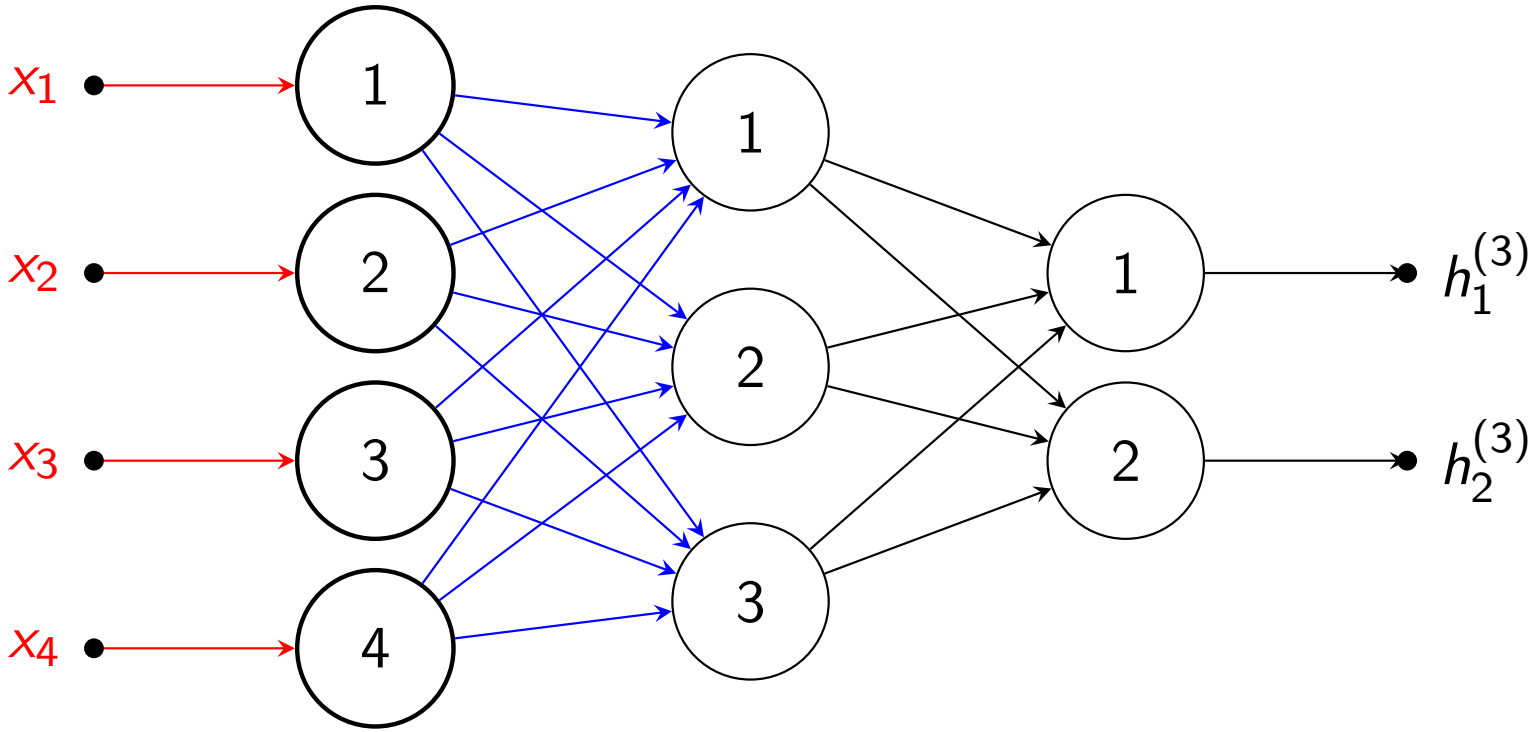


Myötäsyöttöprosessi

- Miten määrätään verkon lähtö (output), kun syöte (input) tunnetaan?
- Käytetään syötekerroksen neuronien lähtöjä seuraavan kerroksen neuronien syötteinä ja lasketaan neuronien lähtö.
- Edetään näin rekursiivisesti lähtökerrokseen asti.

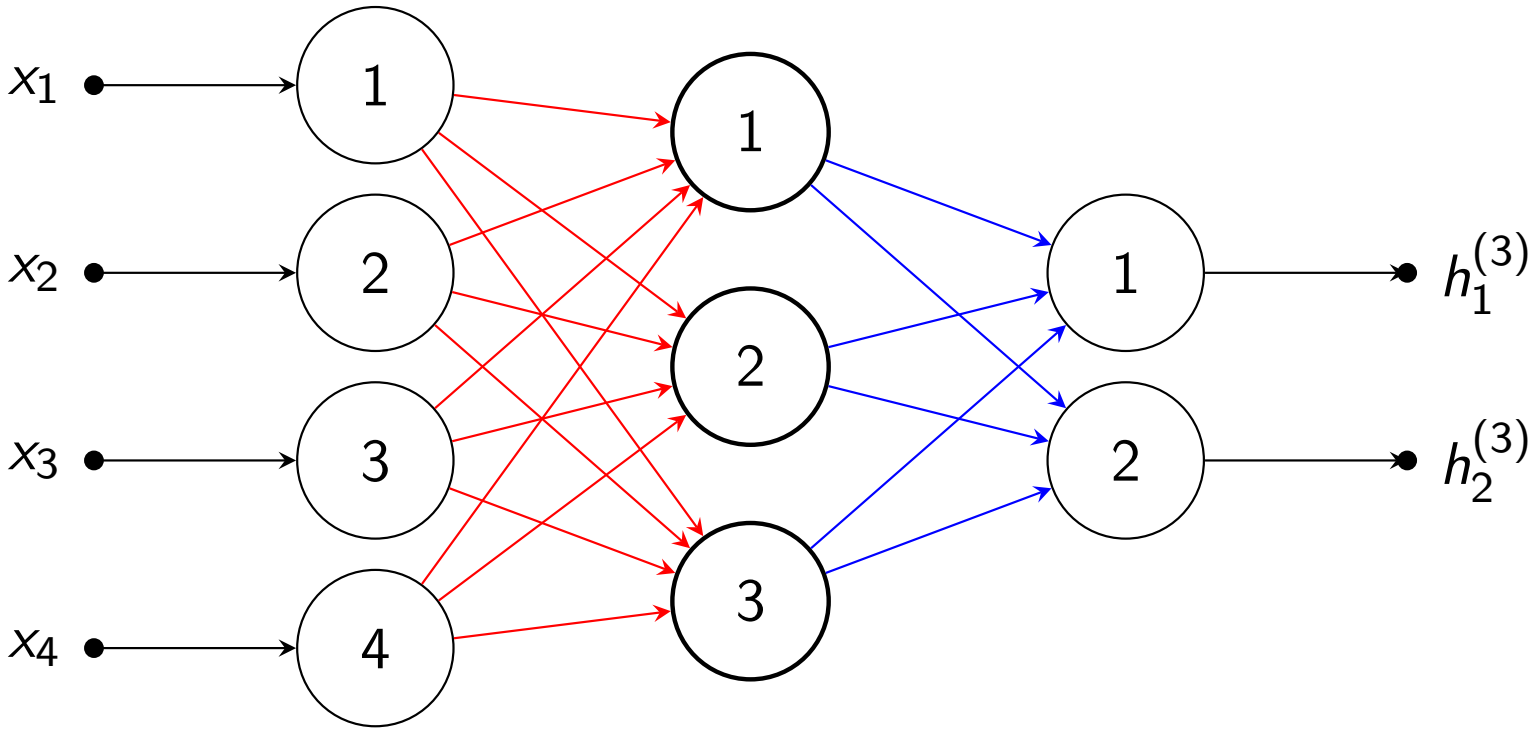
Myötäsyöttöprosessi

$$h^{(1)} = \mathbf{x} = (x_1, x_2, x_3, x_4)$$



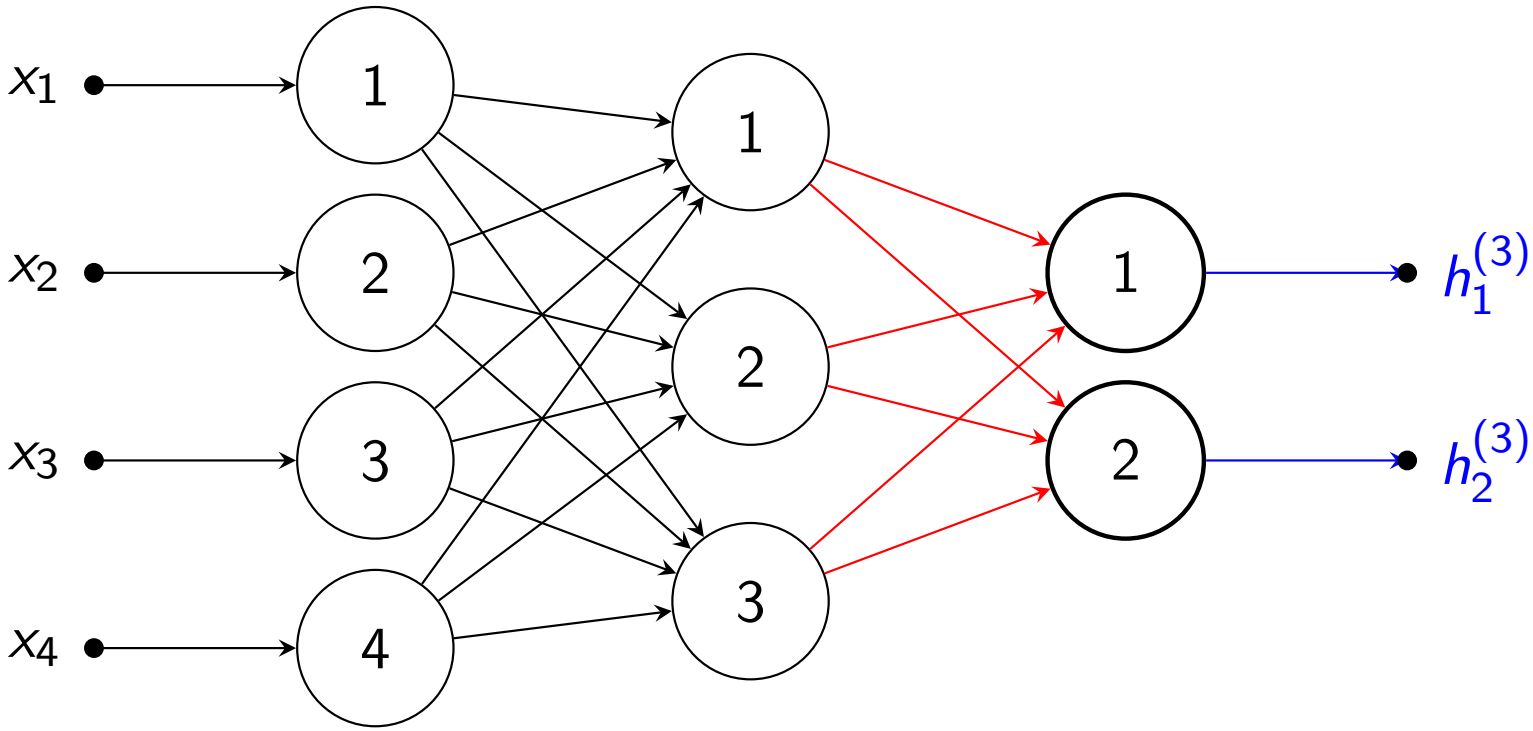
Myötäsyöttöprosessi

$$h^{(2)} = f(z^{(2)}) = f(W^{(1)} h^{(1)} + b^{(1)})$$



Myötäsyöttöprosessi

$$h^{(3)} = f(z^{(3)}) = f(W^{(2)} h^{(2)} + b^{(2)})$$



Neuroverkon opettaminen

- Miten määrätään opetusdatan avulla painot ja bias-termit siten, että verkko approksimoisi mahdollisimman hyvin syötteiden ja lähtöjen välistä riippuvuutta?
- Kvantifioidaan tavoite määrittelemällä aluksi yksittäiselle opetusesimerkille $(x^{(i)}, y^{(i)})$ virhefunktio

$$J(W, b; x^{(i)}, y^{(i)}) = \frac{1}{2}\|h_{W,b}(x^{(i)}) - y^{(i)}\|^2$$

  - kuvaa verkon tuottaman ennusteen ja todellisen arvon välistä eroa

Neuroverkon opettaminen

- Huomioidaan kaikki opetusnäytteet määrittelemällä kokonaisvirhefunktio

$$J(W,b) = \underbrace{\frac{1}{M}\sum_{i=1}^{M} J(W,b;x^{(i)},y^{(i)})}_{\text{keskineliövirhe}} + \underbrace{\frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(w_{ji}^{(l)}\right)^2}_{\text{painotermi}}.$$

- Tavoitteena on etsiä sellaiset W^* ja b^*, että J minimoituu, ts.

$$W^*, b^* = \arg\min J(W,b)$$

$\Rightarrow$ optimointiongelma!

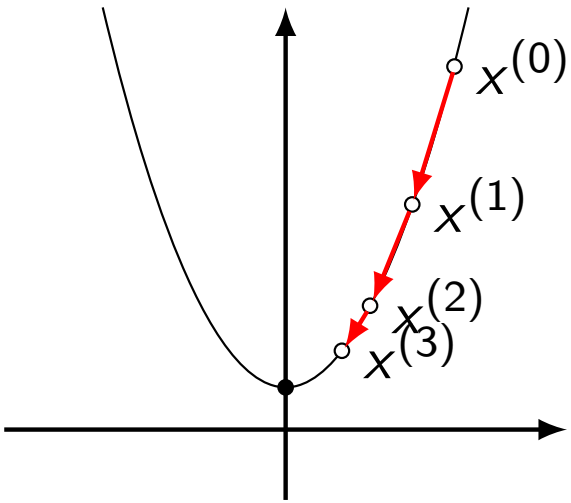
Gradienttilaskeutuminen

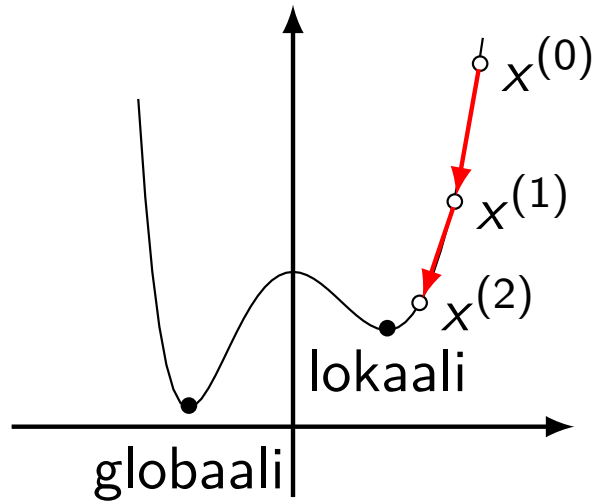
- Päivityssäännöt painoille ja biaksille:

$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \alpha \frac{\partial J(W,b)}{\partial w_{ij}^{(l)}},$$

$$b_i^{(l)} \leftarrow b_i^{(l)} - \alpha \frac{\partial J(W,b)}{\partial b_i^{(l)}}.$$

$$\boxed{x^{(i+1)} = x^{(i)} - \alpha \cdot \nabla f\left(x^{(i)}\right)}$$





Gradienttilaskeutuminen

- Suoralla laskulla havaitaan, että

$$\frac{\partial J(W,b)}{\partial w_{ij}^{(l)}} = \frac{1}{M}\sum_{k=1}^{M} \frac{\partial J(W,b;x^{(k)},y^{(k)})}{\partial w_{ij}^{(l)}} + \lambda w_{ij}^{(l)},$$

$$\frac{\partial J(W,b)}{\partial b_{i}^{(l)}} = \frac{1}{M}\sum_{k=1}^{M} \frac{\partial J(W,b;x^{(k)},y^{(k)})}{\partial b_{i}^{(l)}}.$$

- Miten määrätään osittaisderivaatat

$$\frac{\partial J(W,b;x^{(i)},y^{(i)})}{\partial w_{ij}^{(l)}} \text{ ja } \frac{\partial J(W,b;x^{(i)},y^{(i)})}{\partial b_{i}^{(l)}}?$$

Vastavirta-algoritmi

- Tarkastellaan ensin lähtökerrosta. Voidaan osoittaa, että nyt

$$\frac{\partial J}{\partial w_{12}^{(2)}} = \frac{\partial J}{\partial h_1^{(3)}} \frac{\partial h_1^{(3)}}{\partial z_1^{(3)}} \frac{\partial z_1^{(3)}}{\partial w_{12}^{(2)}} = \underbrace{\left[-(y_1 - h_1^{(3)}) \cdot f'(z_1^{(3)})\right]}_{\delta_1^{(3)}} h_2^{(2)}.$$

- Yleisesti vektorimuodossa lähtökerrokselle

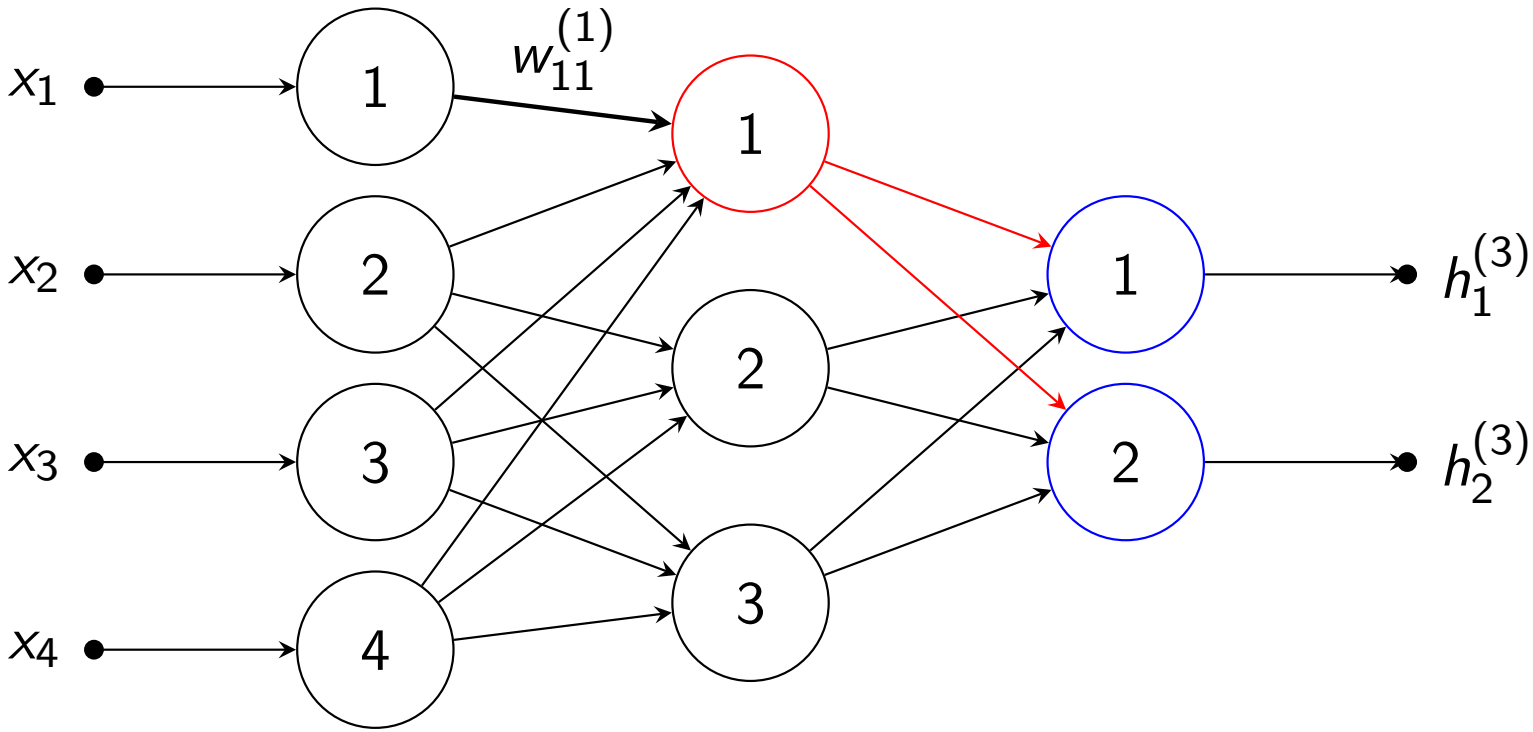
$$\delta^{(n_l)} = -(y^{(i)} - h^{(n_l)}) \odot f'(z^{(n_l)}),$$

jolloin

$$\frac{\partial J}{\partial W^{(n_l-1)}} = \delta^{(n_l)} (h^{(n_l-1)})^\top.$$

Vastavirta-algoritmi

- Havaitaan, että piilokerroksen neuroni vaikuttaa koko verkon lähtöön lähtökerroksen neuronien välityksellä.



Vastavirta-algoritmi

- Edetään piilokerrokseen kirjoittamalla

$$\frac{\partial J}{\partial w_{11}^{(1)}} = \frac{\partial J}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial w_{11}^{(1)}} = \left(\sum_{i=1}^{2} \underbrace{\frac{\partial J}{\partial z_i^{(3)}}}_{\delta_i^{(3)}} \frac{\partial z_i^{(3)}}{\partial z_1^{(2)}} \right) \frac{\partial z_1^{(2)}}{\partial w_{11}^{(1)}}$$

  $\Rightarrow$ laskennassa voidaan hyödyntää edeltävän kerroksen jo laskettuja virhetermejä!

- Voidaan näyttää, että yleisesti piilokerroksille

$$\delta^{(l)} = \left[(W^{(l)})^\top \delta^{(l+1)} \right] \odot f'(z^{(l)}),$$

  jolloin

$$\frac{\partial J}{\partial W^{(l)}} = \delta^{(l+1)} (h^{(l)})^\top.$$

Vastavirta-algoritmi

1. Lasketaan lähtökerrokselle $\delta^{(n_l)} = -(y^{(i)} - h^{(n_l)}) \odot f'(z^{(n_l)})$.
2. Lasketaan rekursiivisesti

$$\delta^{(l)} = \left[(W^{(l)})^\top \delta^{(l+1)}\right] \odot f'(z^{(l)})$$

   ensimmäiseen piilokerrokseen asti.
3. Määrätään osittaisderivaatat painojen suhteeen:

$$\frac{\partial J}{\partial W^{(l)}} = \delta^{(l+1)} (h^{(l)})^\top.$$

4. Määrätään osittaisderivaatat biasten suhteen:

$$\frac{\partial J}{\partial b^{(l)}} = \delta^{(l+1)}.$$

Opetusalgoritmi

Algoritmi 3 (Opetusalgoritmi)

Syöttö: opetusdatajoukko $L = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\}$, painotuskerroin λ, oppimisnopeus α

Askel 1: Alustetaan kaikki painomatriisit $W^{(l)}$ ja biasvektorit $b^{(l)}$ satunnaisluvuilla

Askel 2: Alustetaan $\Delta W^{(l)}$ ja $\Delta b^{(l)}$ nolliksi kaikille l

Askel 3: **jokaiselle** $(x^{(i)}, y^{(i)}) \in L$:

a. Suoritetaan myötäsyöttö syötteellä $x^{(i)}$

b. Lasketaan vastavirta-algoritmilla virhevektorit $\delta^{(l)}$

c. $\Delta W^{(l)} \leftarrow \Delta W^{(l)} + \delta^{(l+1)}(h^{(l)})^\top$ kaikille l

d. $\Delta b^{(l)} \leftarrow \Delta b^{(l)} + \delta^{(l+1)}$ kaikille l

Askel 4: $W^{(l)} \leftarrow W^{(l)} - \alpha \left[\frac{1}{M} \Delta W^{(l)} + \lambda W^{(l)} \right]$

Askel 5: $b^{(l)} \leftarrow b^{(l)} - \alpha \left[\frac{1}{M} \Delta b^{(l)} \right]$

Askel 6: Palataan askeleeseen 2, jos lopetusehtoa ei ole saavutettu

Ulostulo: optimaaliset painomatriisit $W^{(l)}$ ja biasvektorit $b^{(l)}$