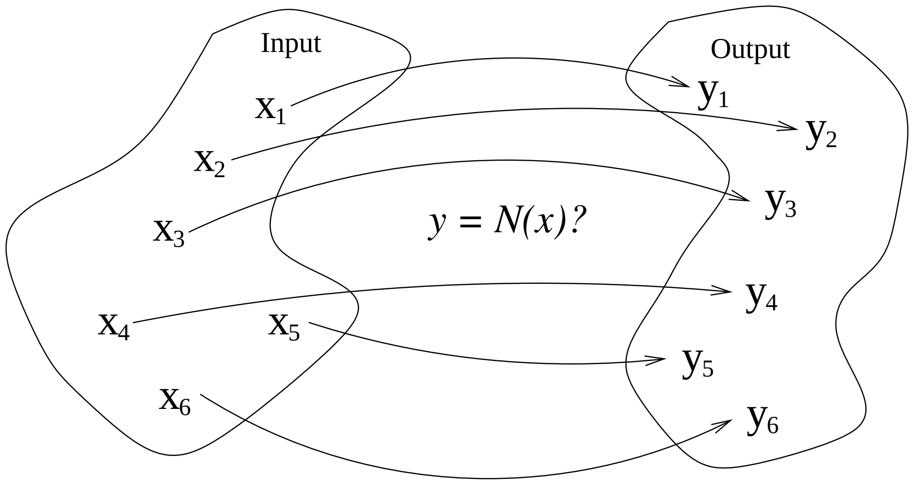
Luku 4

Hermoverkot

Tässä luvussa käydään läpi kurssilla tarvittavat asiat hermoverkoista. Lisätietoja, tarkennuksia ja kohtuullinen joukko erilaisia viitteitä löytyy raportista:

[TK] T. KÄRKKÄINEN, *MLP-network in a layer-wise form: Derivations, consequences and applications to weight decay*, Report No. C 1, University of Jyväskylä, Department of Mathematical Information Technology, 2000.

Hermoverkoista (neural network) on tullut viime vuosina eräs keskeinen elementti ns. laskennallisesti älykkäiden järjestelmien -nimellä (Computationally Intelligent Methods, Soft Computing) kulkevien tekniikoiden joukossa. Peruslähtökohtana on opettaa reaalimaailman dataa hyväksikäyttäen sopivalle tietorakenteelle tuntematon kuvaus, joka liittää mitattujen input-output -muuttujaparien arvot toisiinsa. Tätä on havainnollistettu kuvassa 4.1. Annettuja vektoreita $\{\mathbf{x}_i, \mathbf{y}_i\}$, joiden avulla tuntematon kuvaus määrätään, kutsutaan *opetusaineistoksi*.



Kuva 4.1: Input-Output parien muodostamat joukot.

Opetusaineisto eli opetusdata muodostuu tällä kurssilla joukosta vektoreita

$$\mathbf{x}_i \simeq \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_{n_0} \end{bmatrix} \in \mathbf{R}^{n_0} \quad \text{ja} \quad \mathbf{y}_i \simeq \mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{n_2} \end{bmatrix} \in \mathbf{R}^{n_2} \quad \text{kaikilla } i = 1, \dots, N,$$

missä N on annettujen vektoreiden lukumäärä. Jokaisen vektorin $\mathbf{x}_i$ sisältämiä eri komponentteja $(\mathbf{x}_i)_k,\ k = 1, \ldots, n_0$ kutsutaan *piirteiksi*, missä käytetty nimitys juontaa juurensa yleisesti ns. *hahmontunnistuksen* yhteydessä käytetystä terminologiasta. Koko aineisto $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ voidaan tallettaa kätevästi, yllätys yllätys, matriiseihin:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_N \end{bmatrix} \in \mathbf{R}^{n_0 \times N} \quad \text{ja} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \ldots & \mathbf{y}_N \end{bmatrix} \in \mathbf{R}^{n_2 \times N}.$$

Näin saadaan selkeä ja (ainakin MATLABilla) helposti käsiteltävä talletusmuoto koko annetulle tietomassalle.

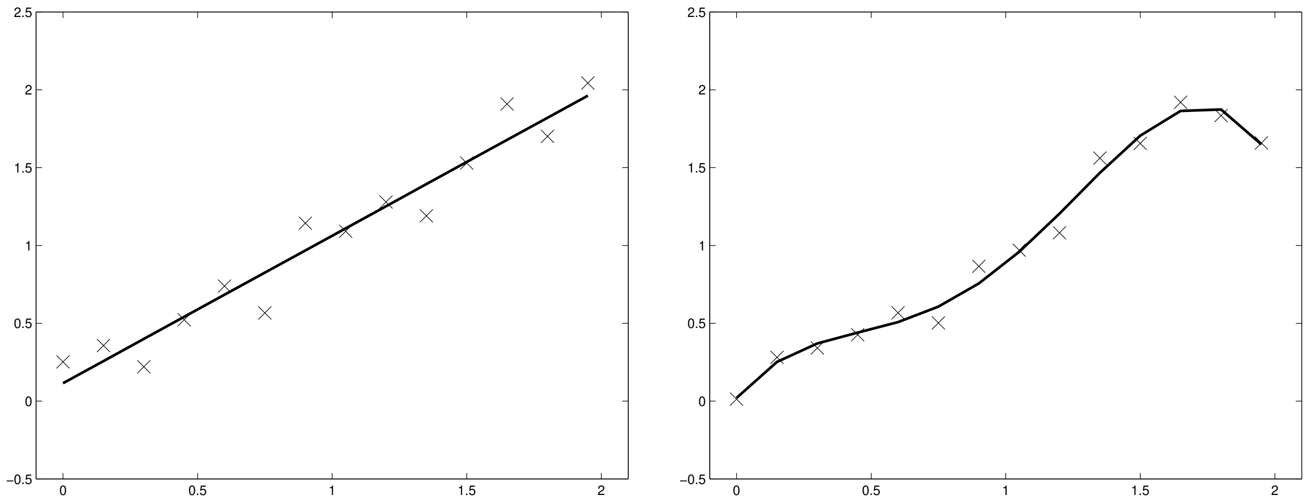
Hermoverkkoja käytetään mitä moninaisimpien ilmiöiden simuloinnissa (eli jäljittelyssä) ja säädössä (eli tilan ohjauksessa). Yleisimpiä sovellusalueita ovat erilaisiin teollisuusprosesseihin ja aikasarjoihin liittyvät seuranta- ja ennustustehtävät (paperitehtaan ohjausjärjestelmä, pörssikurssit jne.) sekä erityyppiset luokittelutehtävät (vaikkapa pullonpalautusautomaatti). Se, miten hyvin erilaisten käytännön sovellusten käsittely verkkojen avulla onnistuu, riippuu keskeisesti kahdesta eri tekijästä: 1. Kuinka hyvin annettu opetusaineisto kuvaa sovelluksen taustalla olevaa tuntematonta systeemiä? 2. Osaammeko konfiguroida eli opettaa verkon oikealla tavalla? Nämä kysymykset eivät liity pelkästään hermoverkkoihin, vaan ne muodostavat myös esimerkiksi yritystaloustieteeseen, ohjelmistotekniikkaan sekä optimointiin liittyvän problematiikan ydinalueen. Tätä voidaan havainnollistaa seuraavilla kysymyksillä:

Yritystaloustiede:	"Are we building the right product?" "Are we building the product right?"
Ohjelmistotekniikka:	"Are we building the right software?" "Are we building the software right?"
Hermoverkot:	"Are we training the right network?" "Are we training the network right?"
Optimointi:	"Are we minimizing the right functional?" "Are we minimizing the functional right?"

Hermoverkkojen käsittely liittää yhteen aikaisemmin jo oppimamme(?) asiat. Verkon rakenne voidaan kuvata yksinkertaisesti ja yksiselitteisesti matriisien avulla. Verkon opettaminen perustuu sopivan optimointitehtävän konstruointiin ja ratkaisemiseen. Tehtävän järkevyyden kannalta tarvitaan annetun opetusdatan esi- ja jälkikäsittelyä. Saatuja ratkaisuja voidaan havainnollistaa ja tarkastella MATLABin graafisia ominaisuuksia hyväksikäyttäen.

Periaatteessa tällä kurssilla käsiteltävä ns. *monikerroksinen Perceptron*-verkko MLP (*Multi-Layered Perceptron*) on vain yksi tapa approksimoida jotain tuntematonta funktiota annetun datan pohjalta. Tilastolliselta kannalta katsottuna kyseessä on siten ns. *epälineaarinen regressio* -analyysi. Kysymys siitä, onko käyttämämme malli tilastolliselta kannalta *parametrinen* vai *ei-parametrinen*, jääköön tilastotieteilijöiden pohdittavaksi (jos em. käsitteet saadaan joskus määriteltyä yksikäsitteisesti). Niille, joiden pohjat riittävät, voidaan todeta, että annetun datan "hyvyys" (ja siten MLP-mallin toimivuus) riippuu siitä, pysyykö *yhteisjakauma* $p(x, y) = p(y|x)p(x)$ samana annetun opetusdatan suhteen.

Yleistä funktion approksimointia on tarkemmin havainnollistettu seuraavassa kuvassa:



Vasemmanpuoleisessa kuvassa on annettun datan lineaarinen approksimaatio, joka saadaan aikaiseksi ratkaisemalla optimointitehtävä

$$\min_{a,b} \sum_{i=1}^{N} |y_i - (ax_i + b)|^2.$$

Oikealla (eri) dataa on approksimoitu neljännen asteen polynomilla, jonka määräämiseksi ratkaistavan optimontitehtävän muotoilu jätetään jokaisen omaksi päänvaivaksi.

Kuvat on tuotettu seuraavanlaisella MATLAB-makrolla, jossa fitattavan polynomin asteluku määrätään muuttujan `deg` avulla:

```
x = 0:0.15:2; e = 6e-1; deg = 1; y = x + e*(rand(size(x))-0.5);
a = plot(x,y,'bx','MarkerSize',14); axis([-0.1 2.1 -0.5 2.5]);
set(gca,'XTick',[0 0.5 1 1.5 2]); p = polyfit(x,y,deg); y2 = polyval(p,x);
hold on, l = plot(x,y2,'g-'); set(l,'LineWidth',2)
```

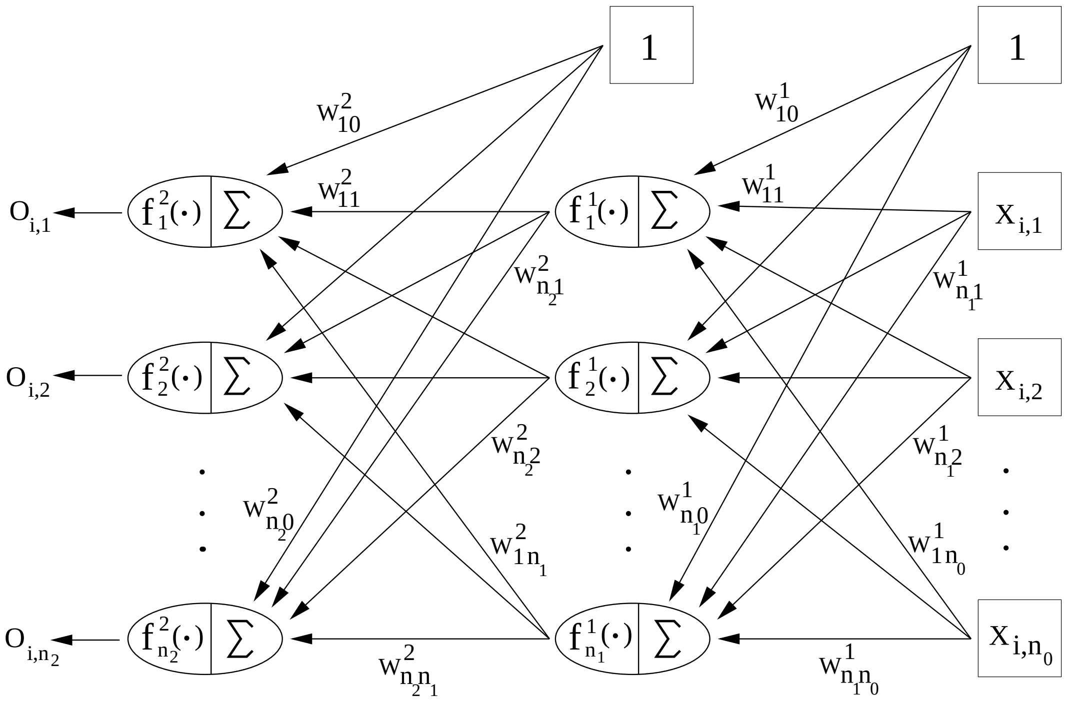
4.1 Miten MLP-verkko muodostuu?

Tarkastellaan aluksi vaihe vaiheelta kuvassa 4.2 havainnollistetun MLP-verkon rakenteen muodostumista. Tavoitteena on ensinnäkin saada jonkinlainen tuntuma siihen kuvaukseen, jonka verkon arkkitehtuuri muodostaa. Yleensä tällaisen verkon toimintaa tulkitaan ihmisaivojen näkökulmasta siten, että rakenne liittää yhteen pienet peruslaskentayksiköt, neuronit. Meille paljon hyödyllisempi esitystapa tulee olemaan algebrallinen muotoilu, jossa verkon toimintaa kuvataan matriisien ja funktioiden avulla, koska tällainen esitysmuoto voidaan esittää sellaisenaan MATLABilla.

Askel 1:

Lähdetään ihan aluksi liikkeelle annettujen lukujen $x_1, \ldots, x_n$ avulla muodostetusta *lineaarisesta* muunnoksesta, joka tapauksessa $n = 1$ määrittelee suoran yhtälön, $n = 2$ tason yhtälön ja tapauksissa $n = 3, 4, \ldots$ ns. hypertason.

$$a(\mathbf{x}) = w_0 + w_1 x_1 + \cdots + w_{n-1} x_{n-1} + w_n x_n = \mathbf{w}^T \hat{\mathbf{x}}, \tag{4.1}$$



Kuva 4.2: Kaksikerroksinen MLP-verkko.

missä

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \quad \text{ja} \quad \hat{\mathbf{x}} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}. \tag{4.2}$$

Tässä lukua w_0 kutsutaan *bias*-termiksi ja sen ansiosta muunnoksen ei välttämättä tarvitse kulkea origon kautta ($a(0) = w_0$). Alkuperäisen vektorin $\mathbf{x}$ laajentaminen muotoon $\hat{\mathbf{x}}$ mahdollistaa kaavassa (4.1) esiintyvän kompaktin (siis lyhyen ja ytimekkään) merkintätavan $\mathbf{w}^T\hat{\mathbf{x}}$. Muunnoksen määrää yksikäsitteisesti *painovektori* $\mathbf{w}$, joka koostuu luvuista $w_0, \ldots, w_n$. Erityisesti, jos $\|\mathbf{w}\| = 1$, reaaliluku $a(\mathbf{x}) = \mathbf{w}^T\hat{\mathbf{x}}$ määrää pituuden vektorin $\hat{\mathbf{x}}$ *projektiolle* $p_{\mathbf{w}}(\hat{\mathbf{x}}) = \mathbf{w}^T\hat{\mathbf{x}}\,\mathbf{w}$ painovektorin $\mathbf{w}$ määräämälle hypertasolle.

Askel 2:

Luonnollisesti samaan vektoriin $\mathbf{x}$ voidaan soveltaa myös useampia lineaarisia muunnoksia:

$$\begin{aligned}
a_1(\mathbf{x}) &= w_{10} + w_{11}x_1 + \cdots + w_{1,n-1}x_{n-1} + w_{1,n}x_n = \mathbf{w}_1^T\hat{\mathbf{x}}, \\
a_2(\mathbf{x}) &= w_{20} + w_{21}x_1 + \cdots + w_{2,n-1}x_{n-1} + w_{2,n}x_n = \mathbf{w}_2^T\hat{\mathbf{x}}, \\
&\vdots \\
a_i(\mathbf{x}) &= w_{i,0} + w_{i,1}x_1 + \cdots + w_{i,n-1}x_{n-1} + w_{i,n}x_n = \mathbf{w}_i^T\hat{\mathbf{x}}, \\
&\vdots \\
a_m(\mathbf{x}) &= w_{m,0} + w_{m,1}x_1 + \cdots + w_{m,n-1}x_{n-1} + w_{m,n}x_n = \mathbf{w}_m^T\hat{\mathbf{x}}.
\end{aligned} \tag{4.3}$$

Nyt huomataan, että asettamalla painovektorit $\mathbf{w}_1, \ldots, \mathbf{w}_m$ matriisin $\mathbf{W}$ riveiksi

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1^T \\ \vdots \\ \mathbf{w}_m^T \end{bmatrix}, \tag{4.4}$$

voidaan kaavassa (4.3) suoritetut $m-$kappaletta muunnoksia esittää kompaktisti muodossa

$$\mathbf{a} = \mathbf{W}\hat{\mathbf{x}}, \tag{4.5}$$

missä vektori $\mathbf{a}$ sisältää saadut reaaliluvut $a_1, \ldots, a_m$. Tämä kaava määrittelee ns. *yksikerroksisen,* lineaarisen Perceptron-verkon datavektorille $\mathbf{x}$ tekemän muunnoksen.

Askel 3:

Useat käytönnön prosesseja kuvaavat ilmiöt eivät kuitenkaan ole lineaarisia vaan *epälineaarisia*, jolloin niitä kuvaavan mallin tulisi myös olla epälineaarinen. Kuinka tämä ominaisuus voitaisiin lisätä lineaariseen muunnokseen $\mathbf{a} = \mathbf{W}\hat{\mathbf{x}}$? No, esimerkiksi soveltamalla jokaiseen saadun vektorin $\mathbf{a}$ komponenttiin a_i jotain "sopivaa" epälineaarista funktiota $f(a_i) = f_i(a_i)$. Tämäntyyppiselle muunnokselle saadaan myös kompakti merkintätapa määrittelemällä ns. *diagonaalinen funktiomatriisi* $\mathcal{F} = \mathcal{F}(\cdot) = \mathrm{Diag}\{f_i(\cdot)\}_{i=1}^m$ muodossa

$$\mathcal{F} = \begin{bmatrix} f_1(\cdot) & 0 & \ldots & 0 \\ 0 & f_2(\cdot) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & f_m(\cdot) \end{bmatrix} \tag{4.6}$$

ja laajentamalla sopivasti matriisi-vektori operaatioiden määritelmiä. Erityisesti yleisen funktiomatriisin $\mathcal{F}$ ja vektorin $\mathbf{v}$ kertolaskua vastaava laskutoimitus määritellään seuraavasti: $\mathbf{y} = \mathcal{F}(\mathbf{v}) \Leftrightarrow y_i = \sum_{j=1}^m f_{ij}(v_j)$ (eli siis funktiomatriisin tapauksessa matriisin alkiolla kertomista vastaa komponenttifunktion soveltaminen).

Tämän hiukkasen hämärän tuntuisen puljauksen seurauksena voidaan tarkastella lineaarisen muunnoksen $\mathbf{a} = \mathbf{W}\hat{\mathbf{x}}$ yleistämistä muotoon

$$\mathcal{F}(\mathbf{a}) = \mathcal{F}(\mathbf{W}\hat{\mathbf{x}}). \tag{4.7}$$

Tässä funktiomatriisin $\mathcal{F}$ valinta diagonaaliseksi vastaa sitä, että vektorin $\mathbf{a}$ jokaista komponenttia a_i muunnetaan erikseen ns. *aktivaatiofunktiolla* $f_i(\cdot)$.

Kuinkas sopivat aktivaatiofunktiot tulisi sitten valita? Tällä kuten koko tähänastisella prosessilla on biologinen taustansa. Nimittäin 60-luvulla psykologit (erityisesti Rosenblatt) tulivat siihen tulokseen, että aivot koostuvat pienistä yksiköistä, neuroneista, jotka niihin liittyvistä muista neuroneista tulevan kokonaisherätteen (jota kuvaa yksinkertaisimmillaan $\mathbf{w}^T\mathbf{x}$) perusteella joko hyväksyvät ja lähettävät herätteensä eteenpäin (aktivoituminen) tai hylkäävät sen. Matemaattisesti tätä voitaisiin kuvata asettamalla

$$f(a) = \begin{cases} 1, & a \geq 0, \\ 0, & a < 0, \end{cases} \tag{4.8}$$

joka määrittelee ns. *askelfunktion*.

Ensimmäiset aivojen toimintaa jäljittelevät hermoverkot perustuivatkin askelfunktion käyttöön aktivaatiossa. Keskeinen tähän valintaan liittyvä ongelma on kuitenkin askelfunktion *ei-differentioituvuus* (derivaattaa klassisessa mielessä ei ole olemassa pisteessä 0), joka vaikeutti olennaisesti tehokkaiden opetusalgoritmien kehittämistä, sillä esimerkiksi jo oppimiamme(?) optimoinnnin perusmenetelmiä ei voitu soveltaa sellaisenaan. Siksipä askelfunktio päätettiinkin korvata jollain sitä "muistuttavalla" sileällä (siis derivoituvalla) vastineella. Matemaattiselta kannalta olennaisia ominaisuuksia sileyden lisäksi ovat aktivaatiofunktion *ei-polynomisuus* ja *monotonisuus*: $f(a) \leq f(b)$ kun $a \leq b$. Monotoonisuuden taustalla on se luonnollinen vaatimus, että aktivaatiossa neuroni ei saa muuttaa saamiensa herätteiden järjestysrelaatiota. Nopeasti käytetyimmäksi aktivaatiofunktioksi (tämä pätee tänäkin päivänä) yleistyi *logistinen sigmoidi*

$$s(a) = \frac{1}{1 + \exp(-a)},$$

joka on pelkästään erikoistapaus yleisempää muotoa olevista funktioista

$$s_k(a) = \frac{1}{1 + \exp(-k\,a)}, \quad k = 1, 2, \ldots \tag{4.9}$$

jotka lähenevät askelfunktiota kun $k \to \infty$. Sigmoidi-tyyppisen aktivaatiofunktion valintaa tukevat myös sekä fysiologiset, aivojen neuroneilla tehdyt mittaukset, että jotkin luokitteluun liittyvät tilastolliset tarkastelut.

Toinen yleistys usein käytetylle aktivaatiofunktiolle on ns. 'k-tanh' funktio

$$t_k(a) = \frac{\exp(k\,a) - \exp(-k\,a)}{\exp(k\,a) + \exp(-k\,a)} = \frac{2}{1 + \exp(-2\,k\,a)} - 1 = 2\,s_k(2\,a) - 1. \tag{4.10}$$

Jatkossa tarvitsemme myös näiden annettujen funktioiden derivaattoja, jotka ovat muotoa

$$s_k^{'}(a) = k\,\exp(k\,a)/(1 + \exp(k\,a))^2 = k\,s_k(a)\,(1 - s_k(a))$$

ja

$$t_k^{'}(a) = 4\,k\,\exp(2\,k\,a)/(1 + \exp(2\,k\,a))^2 = k\,(1 + t_k(a))\,(1 - t_k(a)) = k\,(1 - t_k(a)^2).$$

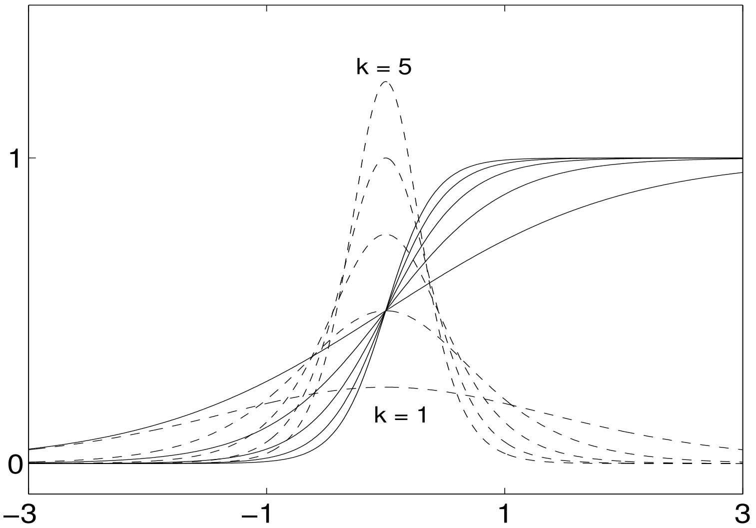
Näistä kaavoista on syytä huomata, että viimeksi annettujen muotojen käyttö laskennassa poistaa mahdollisen nollallajako-ongelman, joka muuten saattaisi syntyä. Funktioita $s_k(a)$ ja vastaavia derivaattoja eri k:n arvoille on havainnollistettu kuvassa 4.3.

Askel 4:

Askeleiden 1–3 avulla on siis saatu aikaan muunnos $\mathcal{F}(\mathbf{W}\hat{\mathbf{x}})$. Mutta tekeekö aktivaatiofunktioiden lisääminen sellaisenaan muunnoksesta varmasti epälineaarisen? Tarkastellaan siis tapausta

$$\mathbf{o} = \mathcal{F}(\mathbf{W}\hat{\mathbf{x}}). \tag{4.11}$$

Koska aktivaatiofunktioilta f_i vaadittiin monotonisuus, on niillä olemassa *käänteisfunktiot* f_i^{-1}, joille siis pätee $f_i(f_i^{-1}(a)) = f_i^{-1}(f_i(a)) = a$. Esimerkiksi logistisen sigmoidin käänteisfunktio on muotoa $g(a) = f^{-1}(a) = \ln(\frac{x}{1-x})$, joka saadaan laskettua identiteetin $f(g(a)) = a$ avulla



Kuva 4.3: Funktiot $s_k(a)$ (viiva) ja $s_k'(a)$ (katkoviiva) arvoille $k = 1, \ldots, 5$.

(tässä voi käyttää apuna esimerkiksi *Mathematica*-ohjelmaa). Vastaavasti yleisemmän funktion $1/(1 + \exp(-ka))$ käänteisfunktio on muotoa $\frac{1}{k}\ln(\frac{x}{1-x})$. Käänteisfunktioista muodostetun diagonaalisen funktiomatriisin $\mathcal{F}^{-1}$ (jolle siis pätee $\mathcal{F}^{-1}(\mathcal{F}(\mathbf{a})) = \mathcal{F}(\mathcal{F}^{-1}(\mathbf{a})) = \mathbf{a}$) avulla kaavaa (4.11) voidaan muokata muotoon

$$\mathbf{o} = \mathcal{F}(\mathbf{W}\hat{\mathbf{x}}) \Leftrightarrow \mathcal{F}^{-1}(\mathbf{o}) = \mathbf{W}\hat{\mathbf{x}}, \tag{4.12}$$

mikä vastaakin pelkästään lineaarista muunnosta laajennetulle datavektorille $\hat{\mathbf{x}}$. Tämän vuoksi verkon toiminnan *epälinearisoimiseksi* täytyy muunnoksia $\mathcal{F}(\mathbf{W} \bullet)$ suorittaa useampia peräkkäin, jotta operaatio sisältäisi varmasti ainakin yhden epälineaarisen aktivaatiotason.

Näin saadaan lopulta aikaiseksi MLP-verkko! Yleisin valinta on käyttää pelkästään kaksikerroksista rakennetta, jolloin verkon tekemä muunnos $\mathcal{N}(\mathbf{x})$ voidaan käytettyjen merkintöjen avulla ilmaista muodossa

$$\mathbf{o} = \mathcal{N}(\mathbf{x}) = \mathcal{F}^2(\mathbf{W}^2\widehat{\mathcal{F}}^1(\mathbf{W}^1\hat{\mathbf{x}})).$$

Jatkossa tarkastelemme kuitenkin hieman yksinkertaistettua muotoilua

$$\mathbf{o} = \mathcal{N}(\mathbf{x}) = \mathbf{W}^2\widehat{\mathcal{F}}(\mathbf{W}^1\hat{\mathbf{x}}), \tag{4.13}$$

joka siis sisältää pelkän ensimmäisen lineaarisen muunnoksen aktivoinnin. Yllä verkon tasoa on merkitty yläindeksin avulla. Lisäksi huomataan, että $\widehat{\mathcal{F}}$ tarkoittaa luonnollisesti sitä, että toisen tason vastaanottama muunnosvektori $\mathcal{F}(\mathbf{W}^1\hat{\mathbf{x}})$ täytyy myös laajentaa 1:llä $\mathbf{W}^2$:n bias-termin aikaansaamiseksi.

MLP-verkon ensimmäistä kerrosta $\mathcal{F}(\mathbf{W}^1\hat{\mathbf{x}})$ kutsutaan yleisesti *piilokerrokseksi*, koska siinä suoritettu operaatio "peittyy" verkon antamassa ulostulossa $\mathcal{N}(\mathbf{x})$ toisen kerroksen muunnoksen alle. Kaksikerroksista rakennetta käytetään toisaalta sen yksinkertaisuuden ja toisaalta sen tosiasian vuoksi, että jo tällaisen rakenteen avulla saadaan muodostettua ns. *universaali aproksimaattori*, joka voi approksimoida *mielivaltaista* kuvausta kahden vektoriavaruuden $\mathbf{R}^n$ ja $\mathbf{R}^m$ välillä (kunhan piilokerroksessa olevien neuronien määrää kasvatetaan äärettömäksi :-).

MLP-verkon toteuttaminen MATLABilla

Tarkastellaan seuraavaksi MATLAB-toteutusta MLP-verkolle sekä verkon tekemän muunnosvektorin $\mathbf{o} = \mathcal{N}(\mathbf{x})$ määräämiselle.

```
% Initialization of the three dimensions:
% n0 for input, n1 for hidden, and n2 for output.
%
n0 = 3; n1 = 4; n2 = 2;
%
% Initialization of the necessary data structures.
%
x = zeros(n0,1); w1 = zeros(n1,n0+1); w2 = zeros(n2,n1+1);
% ...
% It is now assumed that x, w1, and w2 contain reasonable values,
% which have been plugged in after the initialization.
%
o1 = w1*[1; x]; o1 = 1./(1 + exp(-o1)); o = w2*[1; o1];
%
% Isn't it simple?!
%
```

Yllä on siis käytetty aktivaatiofunktiona logistista sigmoidia $f(a) = 1/(1 + exp(-a))$. Yleisemmässä tapauksessa piilokerroksen (laajentamattoman) ulostulon `o1` laskenta korvataan valittujen aktivaatiofunktioiden soveltamisen suorittavan aliohjelman kutsulla. Tästä nähdään erityisesti se, että MLP-muunnoksen johdossa käytetty hankalalta kuulostava termi 'diagonaalinen funktiomatriisi' onkin implementoinnin kannalta 'easy piece of cake'.

4.2 Miten MLP-verkko opetetaan annetun datan avulla?

MLP-verkon tekemän muunnoksen määrittävät siis painomatriisit $\mathbf{W}^1 = (w^1_{ij})$ ja $\mathbf{W}^2 = (w^2_{ij})$ sekä valitut aktivaatiofunktiot funktiomatriisissa $\mathcal{F}$. Koska aktivaatiofunktiot on tapana valita etukäteen (myös niitä voitaisiin joidenkin annettujen funktiomuotojen joukosta hakea tehtäväkohtaisesti), täytyy meidän pystyä määräämään kaikkien komponenttien w^1_{ij} ja w^2_{ij} arvot verkon *konfiguroimiseksi* eli saattamiseksi toimivaksi. Kuinkas tämä tapahtuu?

Tähän käytetään opetusdataa $\{\mathbf{x}_i,\ \mathbf{y}_i\}_{i=1}^N$, $\mathbf{x}_i \in \mathbf{R}^{n_0}$ ja $\mathbf{y}_i \in \mathbf{R}^{n_2}$, jonka avulla MLP-verkko pyritään saattamaan sellaiseen tilaan, että $\mathcal{N}(\mathbf{x}_i) \sim \mathbf{y}_i$ kaikille $i = 1, \ldots, N$. Verkon konfiguraation määräämiseen sovelletusta menetelmästä käytetään nimitystä *opetusalgoritmi*.

Yleisin tapa on ajatella, että MLP-verkon tulisi toimia siten, että keskimäärin virhe $\mathcal{N}(\mathbf{x}_i) - \mathbf{y}_i$ olisi mahdollisimman pieni. Matemaattinen formulointi tälle ajatukselle johtaa painomatriisien $\mathbf{W}^1$ ja $\mathbf{W}^2$ määräämiseen optimointitehtävän

$$\min_{(\mathbf{W}^1, \mathbf{W}^2)} \mathcal{J}(\mathbf{W}^1, \mathbf{W}^2) \tag{4.14}$$

ratkaisuina, missä

$$\mathcal{J}(\mathbf{W}^1, \mathbf{W}^2) = \frac{1}{2N} \sum_{i=1}^{N} \|\mathcal{N}(\mathbf{x}_i) - \mathbf{y}_i\|^2 = \frac{1}{2N} \sum_{i=1}^{N} \|\mathbf{W}^2\, \widehat{\mathcal{F}}(\mathbf{W}^1\, \hat{\mathbf{x}}_i) - \mathbf{y}_i\|^2 \tag{4.15}$$

on perusajatuksen mukaisesti muodostettu kustannusfunktio.

Optimointitehtäville haluttu yleinen ominaisuus on se, että muuttujien arvot liikkuvat *samassa skaalassa*. Mikäli näin ei ole, voi kustannusfunktion gradientti sisältää hyvin eri suuruusluokkaa olevia komponentteja, jolloin gradientti-pohjaisista optimointimenetelmistä tulee epästabiileja (ratkaisu ei päivityksessä muutu jonkin komponentin suhteen mihinkään). Tästä syystä tehtävään (4.14) liittyvän kustannusfunktion (4.15) "järkevöittämiseksi" tarvitaan opetusdatan *esikäsittelyä*.

Data-analyysissä erityyppiset esi- ja jälkikäsittelyt ovat hyvin yleisiä, ja niitä perustellaan mitä monimaisimmilla tavoilla. Meidän tapauksessamme lähtökohtana on syöttää painomatriiseille $\mathbf{W}^1$ ja $\mathbf{W}^2$ vektoreita, jotka sisältävät samansuuruisia komponentteja. Tämä saadaan aikaan esikäsittelemällä annetun datan $\{\mathbf{X}, \mathbf{Y}\}$ *jokainen piirre* $\mathbf{X}_{k,:}$, $k = 1, \ldots, n_0$ ja $\mathbf{Y}_{k,:}$, $k = 1, \ldots, n_2$ välille $[0, 1]$ (k-sigmoidi aktivointi) tai $[-1, 1]$ (k-tanh aktivointi). Esikäsittelyn varsinainen toteutus saadaan kehittämällä yksinkertainen lineaarinen muunnos, joka kuvaa annetun välin $[a, b]$ (piirteen minimi- ja maksimiarvot) halutulle välille $[c, d]$. Haluttu muunnos $p(x) = \alpha x + \beta$ määräytyy yksikäsitteisesti ehtojen

$$p(a) = c \quad \text{ja} \quad p(b) = d$$

avulla. Saadun verkon todellisissa tilanteissa tapahtuvaa käyttöä varten tarvitaan myös vastaava käänteismuunnos (käänteiskuvaus), joka palauttaa verkon läpi kulkeneen, skaalatun datan alkuperäiseen suuruuteensa.