4.2.1 MLP-verkon optimaalisuusehdot

Kuten luvussa 3 opittiin, gradientti-pohjaisten optimointimenetelmien käyttämiseksi kustannusfunktion (4.14) minimointiin tarvitaan lausekkeet funktion derivaatoille $\frac{\partial \mathcal{J}}{\partial w^1_{ij}}$ ja $\frac{\partial \mathcal{J}}{\partial w^2_{ij}}$. Nimittäin, tehtävän (4.15) lokaalin ratkaisun $(\mathbf{W}^{1^*}, \mathbf{W}^{2^*})$ karakterisoivat ehdot

$$\nabla_{(\mathbf{W}^1,\mathbf{W}^2)}\mathcal{J}(\mathbf{W}^{1^*}, \mathbf{W}^{2^*}) = \begin{bmatrix} \nabla_{\mathbf{W}^1}\mathcal{J}(\mathbf{W}^{1^*}, \mathbf{W}^{2^*}) \\ \nabla_{\mathbf{W}^2}\mathcal{J}(\mathbf{W}^{1^*}, \mathbf{W}^{2^*}) \end{bmatrix} = \begin{bmatrix} \mathbf{O} \\ \mathbf{O} \end{bmatrix}, \tag{4.16}$$

jotka on käytetyn formalismin perusteella annettu jo valmiiksi matriisimuodossa (siis suoraan matriisien $\mathbf{W}^1$ ja $\mathbf{W}^2$ eikä niiden komponenttien avulla).

Seuraavaksi tarkoituksena on johtaa matriisimuotoiset kaavat kustannusfunktion $\mathcal{J}(\mathbf{W}^1, \mathbf{W}^2)$ derivaatoille. Derivaattojen muodostamiseen käytetyt laskutekniikat *eivät ole ollenkaan olennaisia tämän kurssin kannalta*[1], mutta saatuja tuloksia tarvitaan jatkossa.

Allaoleva analyysi edellyttää, että aktivaatiofunktiot funktiomatriisissa $\mathcal{F}$ ovat *derivoituvia*. Aikaisemmin esitettyjen 'k-sig' ja 'k-tanh' funktioiden tapauksessa tämä oletus on luonnollisesti voimassa.

Pohjustetaan optimaalisuusehtojen muodostumista muutaman aputuloksen kautta.

Lemma 1. Olkoot $\mathbf{v} \in \mathbf{R}^{m_1}$ ja $\mathbf{y} \in \mathbf{R}^{m_2}$ annettuja vektoreita. Gradientti-matriisi $\nabla_{\mathbf{W}} J(\mathbf{W}) \in \mathbf{R}^{m_2 \times m_1}$ funktiolle

$$J(\mathbf{W}) = \frac{1}{2}\,\|\mathbf{W}\,\mathbf{v} - \mathbf{y}\|^2$$

on muotoa

$$\nabla_{\mathbf{W}} J(\mathbf{W}) = [\mathbf{W}\,\mathbf{v} - \mathbf{y}]\,\mathbf{v}^T.$$

Todistus. Kustannusfunktio $J(\mathbf{W})$ on komponenteittain aukikirjoitettuna muotoa

$$J(\mathbf{W}) = \frac{1}{2}\sum_{i=1}^{m_2}\left(\sum_{j=1}^{m_1} w_{ij}\,v_j - y_i\right)^2, \tag{4.17}$$

missä i on siis matriisin $\mathbf{W}$ rivi- ja j sarakeindeksi. Nyt suoralla laskulla saadaan

$$\begin{aligned} \frac{\partial J}{\partial w_{ik}} &= \left(\sum_{j=1}^{m_1} w_{ij}\,v_j - y_i\right) v_k \\ \Longrightarrow \frac{\partial J}{\partial \mathbf{w}_{i,:}} &= \left(\sum_{j=1}^{m_1} w_{ij}\,v_j - y_i\right) \mathbf{v}^T = [\mathbf{W}\,\mathbf{v} - \mathbf{y}]_i\,\mathbf{v}^T \\ \Longrightarrow \frac{\partial J}{\partial \mathbf{W}} &= [\mathbf{W}\,\mathbf{v} - \mathbf{y}]\,\mathbf{v}^T. \end{aligned} \tag{4.18}$$

Tässä käytettiin MATLAB-tyylistä lyhennysmerkintää : rivivektorille $\mathbf{w}_{i,:}$. Tämä todistaa tuloksen. □

Lemma 2. Olkoot $\mathbf{W} \in \mathbf{R}^{m_2 \times m_1}$ annettu matriisi, $\mathbf{y} \in \mathbf{R}^{m_2}$ annettu vektori ja $\mathcal{F} = \mathrm{Diag}\{f_i(\cdot)\}_{i=1}^{m_1}$ annettu diagonaalinen funktiomatriisi. Funktion

$$J(\mathbf{u}) = \frac{1}{2}\,\|\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}\|^2, \tag{4.19}$$

gradienttivektori $\nabla_{\mathbf{u}} J(\mathbf{u}) \in \mathbf{R}^{m_1}$ on muotoa

$$\nabla_{\mathbf{u}} J(\mathbf{u}) = \mathrm{Diag}\{\mathcal{F}^{'}(\mathbf{u})\}\,\mathbf{W}^T\,[\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}].$$

[1] EI TAATUSTI TULE TENTTIIN!!!

Todistus. Tarkastellaan jälleen aluksi kustannusfunktiota komponenteittain aukikirjoitettuna

$$J(\mathbf{u}) = \frac{1}{2}\sum_{j=1}^{m_2}\left(\sum_{i=1}^{m_1} w_{ji}\, f_i(u_i) - y_j\right)^2 = \sum_{j=1}^{m_2} J_j(\mathbf{u}), \tag{4.20}$$

missä siis asetettiin $J_j(\mathbf{u}) = 1/2\sum_{i=1}^{m_1}(w_{ji}\, f_i(u_i) - y_j)^2 = 1/2[\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}]_j^2$. Tästä saadaan edelleen laskettua

$$\frac{\partial J_j}{\partial u_k} = \left[\sum_{i=1}^{m_1} w_{ji}\, f_i(u_i) - y_j\right] w_{jk}\, f_k^{'}(u_k) = w_{jk}\, f_k^{'}(u_k)\,[\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}]_j. \tag{4.21}$$

Muistetaan, että $\mathbf{u}$ on vektori, jossa on m_2-riviä. Kuten Lemmassa 1 saadaan $J(\mathbf{u})$:n derivaatta koko vektorin $\mathbf{u}$ suhteen muodostettua käsittelemällä indeksiä k rivi-indeksinä ja indeksiä j sarakeindeksinä. Tällöin saadaan

$$\begin{aligned}\nabla_{\mathbf{u}} J(\mathbf{u}) &= \begin{pmatrix} w_{11}\, f_1^{'}(u_1) & w_{21}\, f_1^{'}(u_1) & \dots & w_{m_2 1}\, f_1^{'}(u_1) \\ w_{12}\, f_2^{'}(u_2) & w_{22}\, f_2^{'}(u_2) & \dots & w_{m_2 2}\, f_2^{'}(u_2) \\ \vdots & \vdots & \ddots & \vdots \\ w_{1m_1}\, f_{m_1}^{'}(u_{m_1}) & w_{2m_1}\, f_{m_1}^{'}(u_{m_1}) & \dots & w_{m_2 m_1}\, f_{m_1}^{'}(u_{m_1}) \end{pmatrix} [\mathbf{W}\mathcal{F}(\mathbf{u}) - \mathbf{y}] \\ &= \left(\mathbf{W}\,\mathrm{Diag}\{\mathcal{F}^{'}(\mathbf{u})\}\right)^T [\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}] = \mathrm{Diag}\{\mathcal{F}^{'}(\mathbf{u})\}\,\mathbf{W}^T\,[\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}],\end{aligned} \tag{4.22}$$

joka on täsmälleen haluttu tulos.

Korollaari 1. Jos lemmassa 2 funktiomatriisi $\mathcal{F}(u)$ on yleistä muotoa (ei välttämättä diagonaalinen), saadaan kustannusfunktion (4.19) gradienttivektorille $\nabla_{\mathbf{u}} J(\mathbf{u}) \in \mathbf{R}^{m_1}$ lauseke

$$\nabla_{\mathbf{u}} J(\mathbf{u}) = \left(\mathbf{W}\,\mathcal{F}^{'}(\mathbf{u})\right)^T [\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}],$$

missä $\mathcal{F}^{'}(\mathbf{u})$ tarkoittaa matriisia, jossa funktioiden derivaattojen arvot on laskettu vektorin $\mathbf{u}$ komponenteille.

Lemma 3. Olkoon $\bar{\mathbf{W}} \in \mathbf{R}^{m_2 \times m_1}$ annettu matriisi, $\mathcal{F} = \mathrm{Diag}\{f_i(\cdot)\}_{i=1}^{m_1}$ annettu diagonaalinen funktio-matriisi ja $\mathbf{v} \in \mathbf{R}^{m_0}$, $\mathbf{y} \in \mathbf{R}^{m_2}$ annettuja vektoreita. Gradientti-matriisi $\nabla_{\mathbf{W}} J(\mathbf{W}) \in \mathbf{R}^{m_1 \times m_0}$ kustannusfunktiolle

$$J(\mathbf{W}) = \frac{1}{2}\,\|\bar{\mathbf{W}}\,\mathcal{F}(\mathbf{W}\mathbf{v}) - \mathbf{y}\|^2 \tag{4.23}$$

on muotoa

$$\nabla_{\mathbf{W}} J(\mathbf{W}) = \mathrm{Diag}\{\mathcal{F}^{'}(\mathbf{W}\mathbf{v})\}\,\bar{\mathbf{W}}^T\,[\bar{\mathbf{W}}\,\mathcal{F}(\mathbf{W}\mathbf{v}) - \mathbf{y}]\,\mathbf{v}^T.$$

Todistus. Jotta laskut helpottuisivat, esitellään aluksi uusi, ylimääräinen muuttuja $\mathbf{u} = \mathbf{W}\mathbf{v}$. Sen sijaan, että käsiteltäisiin suoraan tehtävää (4.23) käsitelläänkin sen kanssa ekvivalenttia (yhtäpitävää), rajoitettua optimointitehtävää

$$\min_{(\mathbf{u},\mathbf{W})} \tilde{J}(\mathbf{u},\mathbf{W}) = \frac{1}{2}\,\|\bar{\mathbf{W}}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}\|^2 \quad \text{ehdolla}\ \ \mathbf{u} = \mathbf{W}\,\mathbf{v}, \tag{4.24}$$

missä $\mathbf{u}$:ta ja $\mathbf{W}$:tä käsitellään toisistaan *riippumattomina* muuttujina, joita annettu rajoite sitoo yhteen. Tehtävään (4.24) liittyvä ns. Lagrange-funktio on muotoa

$$\mathcal{L}(\mathbf{u},\mathbf{W},\boldsymbol{\lambda}) = \tilde{J}(\mathbf{u},\mathbf{W}) + \boldsymbol{\lambda}^T(\mathbf{u} - \mathbf{W}\,\mathbf{v}), \tag{4.25}$$

missä $\boldsymbol{\lambda} \in \mathbf{R}^{m_1}$ sisältää annettuun rajoitteeseen liittyvät ns. Lagrangen kertoimet.
Kun huomataan, että funktionaalit $\tilde{J}(\mathbf{u},\mathbf{W})$ ja $\mathcal{L}(\mathbf{u},\mathbf{W},\boldsymbol{\lambda})$ saavat täsmälleen saman arvon kun rajoite $\mathbf{u} = \mathbf{W}\,\mathbf{v}$ on voimassa, seuraa tästä, että tehtävän (4.24) ratkaisu voidaan karakterisoida myös ns. satulapiste-ehtojen $\nabla\mathcal{L}(\mathbf{u},\mathbf{W}^1,\boldsymbol{\lambda}) = 0$ avulla. Lasketaanpas siis derivaatat $\nabla_{\mathbf{u}}\mathcal{L}$, $\nabla_{\mathbf{W}}\mathcal{L}$ ja $\nabla_{\boldsymbol{\lambda}}\mathcal{L}$ (vektori- ja matriisi-muotoisina) Lagrange-funktiolle (4.25) seuraavaksi.
Gradientti-vektorilla $\nabla_{\mathbf{u}}\mathcal{L}(\mathbf{u},\mathbf{W},\boldsymbol{\lambda})$ on Lemman 2 perusteella esitys

$$\nabla_{\mathbf{u}}\mathcal{L} = \nabla_{\mathbf{u}}\tilde{J}(\mathbf{u}) + \boldsymbol{\lambda} = \mathrm{Diag}\{\mathcal{F}'(\mathbf{u})\}\,\bar{\mathbf{W}}^T\,[\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}] + \boldsymbol{\lambda}. \tag{4.26}$$

Muille derivaatoille saadaan samantyyppisillä laskuilla kuin edelläkin esitykset

$$\begin{aligned} \nabla_{\mathbf{W}}\mathcal{L} &= -\boldsymbol{\lambda}\,\mathbf{v}^T, & (4.27)\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L} &= \mathbf{u} - \mathbf{W}\mathbf{v}. & (4.28) \end{aligned}$$

Käyttämällä (4.26):n mukaista kaavaa $-\boldsymbol{\lambda} = \mathrm{Diag}\{\mathcal{F}'(\mathbf{u})\}\,\bar{\mathbf{W}}^T\,[\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}]$ ja sijoittamalla tämä (4.27):ään saadaan

$$\nabla_{\mathbf{W}}\mathcal{L} = \mathrm{Diag}\{\mathcal{F}'(\mathbf{u})\}\,\bar{\mathbf{W}}^T\,[\bar{\mathbf{W}}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}]\,\mathbf{v}^T. \tag{4.29}$$

Kun sitten $\mathbf{u}$ korvataan (4.29):ssä $\mathbf{Wv}$:llä, saadaan lopulta

$$\nabla_{\mathbf{W}}J = \nabla_{\mathbf{W}}\mathcal{L} = \mathrm{Diag}\{\mathcal{F}'(\mathbf{W}\,\mathbf{v})\}\,\bar{\mathbf{W}}^T\,[\bar{\mathbf{W}}\,\mathcal{F}(\mathbf{W}\,\mathbf{v}) - \mathbf{y}]\,\mathbf{v}^T, \tag{4.30}$$

jossa siis esiintyy pelkästään tuntematon matriisi $\mathbf{W}$. Tehtävien (4.23), (4.24) ja (4.25) yhtäpitävyyden nojalla (4.30) antaa halutun gradientti-matriiisin tehtävälle (4.23). Huh huh! □

Lause 5. Gradienttimatriisit $\nabla_{\mathbf{W}^1}\mathcal{J}(\mathbf{W}^1,\,\mathbf{W}^2)$ ja $\nabla_{\mathbf{W}^2}\mathcal{J}(\mathbf{W}^1,\,\mathbf{W}^2)$ kustannusfunktiolle (4.14) ovat muotoa

(i)
$$\begin{aligned} \nabla_{\mathbf{W}^1}\mathcal{J}(\mathbf{W}^1,\,\mathbf{W}^2) &= \frac{1}{N}\sum_{i=1}^{N}\mathrm{Diag}\{\mathcal{F}'(\mathbf{W}^1\hat{\mathbf{x}}_i)\}\,(\mathbf{W}^2_1)^T\,[\mathbf{W}^2\,\widehat{\mathcal{F}}(\mathbf{W}^1\,\hat{\mathbf{x}}_i) - \mathbf{y}_i]\,\hat{\mathbf{x}}_i^T \\ &= \frac{1}{N}\sum_{i=1}^{N}\mathrm{Diag}\{\mathcal{F}'(\mathbf{W}^1\,\hat{\mathbf{x}}_i)\}\,(\mathbf{W}^2_1)^T\,\mathbf{e}_i\,\hat{\mathbf{x}}_i^T, \end{aligned}$$

(ii)
$$\begin{aligned} \nabla_{\mathbf{W}^2}\mathcal{J}(\mathbf{W}^1,\,\mathbf{W}^2) &= \frac{1}{N}\sum_{i=1}^{N}[\mathbf{W}^2\,\widehat{\mathcal{F}}(\mathbf{W}^1\,\hat{\mathbf{x}}_i) - \mathbf{y}_i]\,[\widehat{\mathcal{F}}(\mathbf{W}^1\,\hat{\mathbf{x}}_i)]^T \\ &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{e}_i\,[\widehat{\mathcal{F}}(\mathbf{W}^1\,\hat{\mathbf{x}}_i)]^T. \end{aligned}$$

Kohdassa (i) $\mathbf{W}^2_1$ on alimatriisi $\left(\mathbf{W}^2\right)_{i,j}$, $i = 1,\ldots,n_2;\ j = 1,\ldots,n_1$, eli matriisi $\mathbf{W}^2$, josta on poistettu bias-termit sisältävä 1.sarake $\mathbf{w}^2_0$ (MATLABissa $\mathbf{W}^2_1 =$ `w2(:,2:n1+1)`).

Todistus. Kohta (ii) seuraa suoraan lemmasta 1.
Laajennus-operaattorin $\widehat{\ }$ määritelmästä seuraa, että kaikille $1 \le i \le N$ pätee

$$\mathbf{e}_i = \mathbf{W}^2\,\widehat{\mathcal{F}}(\mathbf{W}^1\,\hat{\mathbf{x}}_i) - \mathbf{y}_i = [\mathbf{W}^2_0\,\mathbf{W}^2_1]\begin{bmatrix}1\\ \mathcal{F}(\mathbf{W}^1\,\hat{\mathbf{x}}_i)\end{bmatrix} - \mathbf{y}_i = \mathbf{W}^2_0 + \mathbf{W}^2_1\,\mathcal{F}(\mathbf{W}^1\,\hat{\mathbf{x}}_i) - \mathbf{y}_i. \tag{4.31}$$

Kun tätä kaavaa käytetään yhdessä Lemman 3 tuloksen kanssa komponenteittain (siis erikseen jokaiselle $i = 1,\ldots,N$) kustannusfunktioon (4.15), saadaan kohdassa (ii) esitetty kaava välittömästi. □

Edellä esitetyllä tekniikalla on mahdollista laskea gradientti-matriisien kaavat myös useampikerroksiselle MLP-verkolle. Täsmälliset esitykset löytyvät lähteestä [TK].

Päätetään tämä kaavapläjäytys seuraavaan tulokseen, joka valaisee hieman sitä lopullista rakennetta, jonka MLP-verkko datasta oikeasti oppii.

Korollaari 2. MLP-verkon tekemä keskimääräinen virhe $\frac{1}{N}\sum_{i=1}^{N}\mathbf{e}_i^*$ on nolla.

Todistus. Optimaalisuusehto $\frac{1}{N}\sum_{i=1}^{N}\mathbf{e}_i^*\,[\hat{\mathbf{o}}_i^{(1)}]^T = \mathbf{O}$ (missä $o_i^{(1)} = \mathcal{F}(\mathbf{W}^{1*}\hat{\mathbf{x}}_i)$), joka saadaan edellisen lauseen kohdasta (ii), voidaan lausua myös laajentamattomassa muodossa

$$\frac{1}{N}\sum_{i=1}^{N}\mathbf{e}_i^*\,[1\ (\mathbf{o}_i^{(1)})^T] = \mathbf{O}. \tag{4.32}$$

Kun tästä otetaan transpoosi, saadaan

$$\frac{1}{N}\sum_{i=1}^{N}\begin{bmatrix}1\\ \mathbf{o}_i^{(1)}\end{bmatrix}(\mathbf{e}_i^*)^T = \frac{1}{N}\sum_{i=1}^{N}\begin{bmatrix}(\mathbf{e}_i^*)^T\\ \mathbf{o}_i^{(1)}\,(\mathbf{e}_i^*)^T\end{bmatrix} = \begin{bmatrix}\mathbf{O}\\ \mathbf{O}\end{bmatrix}. \tag{4.33}$$

Mutta nyt ensimmäinen rivi kaavassa (4.33) osoittaa, että $\frac{1}{N}\sum_{i=1}^{N}\mathbf{e}_i^* = \mathbf{O}$, mitä haluttiinkin osoittaa. □

Huomautus 1. Huomataan, että ylläoleva tulos oli pelkästään sen seurausta, että piilokerroksen ulostuloon $\mathbf{o}_i^{(1)}$ liitettiin $\mathbf{W}^L$:ään bias-termin lisäävä laajennus vakiolla yksi. Erityisesti sen muunnoksen muodolla, joka tapahtui ennen viimeistä kerrosta, ei ollut saatuun tulokseen mitään vaikutusta.

Kustannusfunktio $\mathcal{J}(\mathbf{W}^1,\mathbf{W}^2)$ ja gradienttimatriisit $\mathbf{DW}^1 = \nabla_{\mathbf{W}^1}\mathcal{J}(\mathbf{W}^1,\mathbf{W}^2)$ ja $\mathbf{DW}^2 = \nabla_{\mathbf{W}^2}\mathcal{J}(\mathbf{W}^1,\mathbf{W}^2)$ voidaan muodostaa annetulle datajoukolle $\{\mathbf{x}_i,\mathbf{y}_i\}_{i=1}^N$ seuraavan algoritmin mukaisesti. Tässä oletetaan, että meillä on olemassa m-funktio `n_net_act`, joka kutsulla `[o,o1,d1] = n_net_act(w2,w1,k,x)` palauttaa MLP-verkon ulostulon `o` sekä piilokerroksen ulostulon `o1` ja vastaavan derivaattavektorin `d1` annetulle vektorille `x` sekä verkon arkkitehtuurin (rakenteen) määrittävälle kolmikolle `w2,w1,k`.

For $i = 1,\ldots,N$

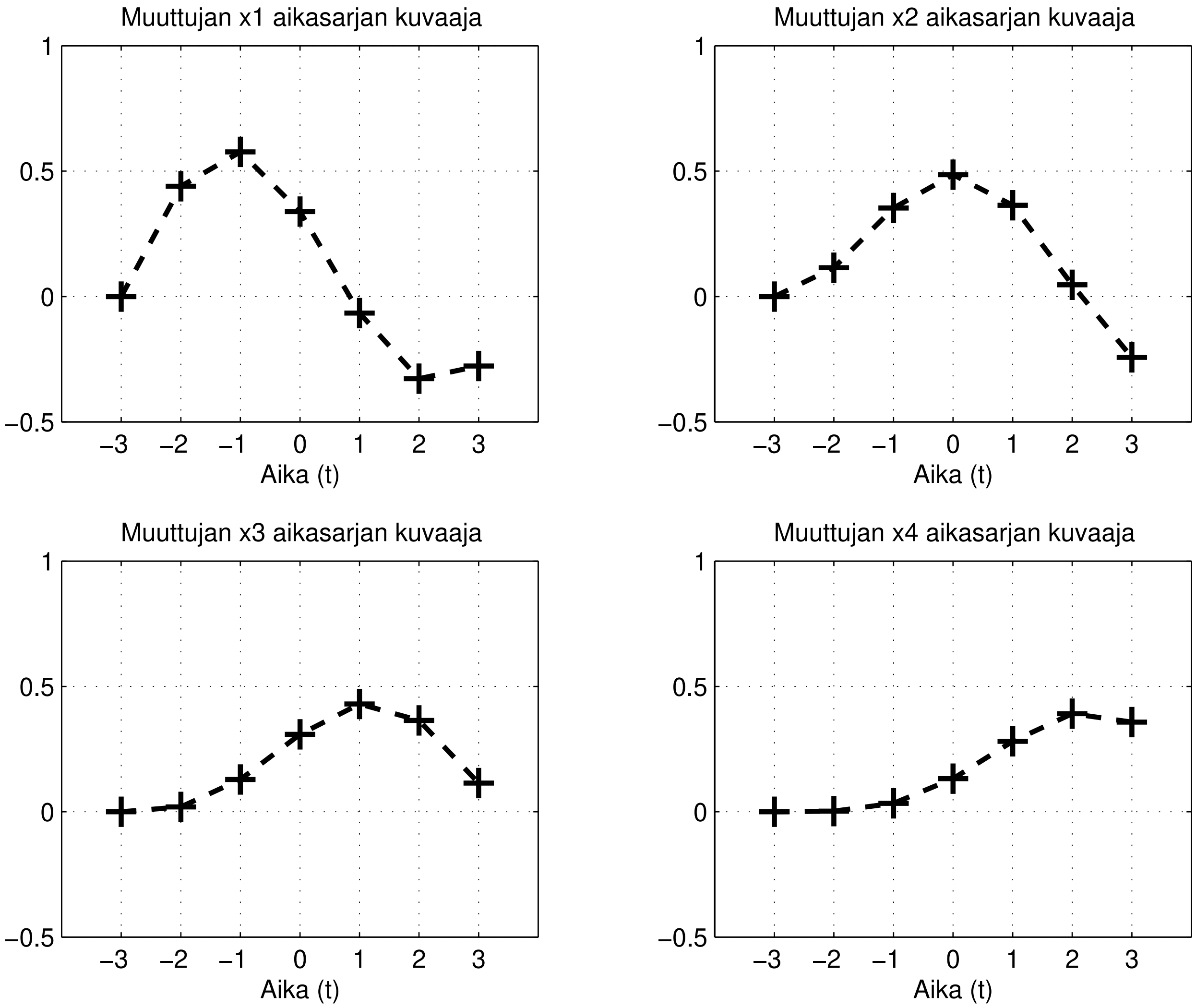
1. Laske verkon muunnos $[\mathbf{e},\mathbf{o},\mathbf{d}] =$ `n_net_act` $(\mathbf{W}^2,\mathbf{W}^1,k,\mathbf{x}_i)$.
2. Laske virhe $\mathbf{e} = \mathbf{e} - \mathbf{y}_i$.
3. Aseta $\mathcal{J} = \mathcal{J} + \frac{1}{2N}\mathbf{e}^T\mathbf{e}$.
   Aseta $\mathbf{DW}^1 = \mathbf{DW}^1 + \frac{1}{N}\,\mathrm{Diag}\{\mathbf{d}\}\,(\mathbf{W}_1^2)^T\,\mathbf{e}\,[1\ \mathbf{x}_i^T]$
   Aseta $\mathbf{DW}^2 = \mathbf{DW}^2 + \frac{1}{N}\,\mathbf{e}\,[1\mathbf{o}^T]$.

End For

Ihan lopuksi muistutetaan mieliin vielä se tosiasia, että gradienttimatriisien nollakohdan hakeminen antaa verkon painoille *lokaalisti* parhaat mahdolliset arvot. Jos halutaan etsiä *globaalia* minimimiä opetukseen käytettävälle kustannusfunktiolle (4.14), täytyy gradienttimenetelmään lisätä jonkinlainen *globalisointi*-strategia. Yksinkertaisimmillaan tämä saadaan aikaiseksi ratkaisemalla MLP-verkon opetustehtävä useita kertoja lähtien satunnaisesti valituista eri aloituspisteistä.

4.3 Esimerkkejä MLP-verkon käytöstä

4.3.1 Aikasarjojen ennustaminen



Kuva 4.4: Muuttujien $x1$–$x4$ aikasarjoja.

MLP-verkon avulla tapahtuvassa aikasarjojen ennustamisessa pyritään verkolle opettamaan tuntematon kuvaus, joka liittää valittujen muuttujien arvot ajanhetkillä $t = 0, -1, \ldots$ (siis nykyhetkellä t ja sitä ennen toteutuneilla arvoilla) johonkin halutun muuttujan/muuttujien arvoon tulevana ajanhetkenä $t + 1$ tai pidemmälle tulevaisuuteen ajanhetkiin $t + 2$, $t + 3, \ldots$. Tekniikka konkretisoituu parhaiten tarkastelemalla kuvassa 4.4 havainnollistettuja neljän muuttujan $x1$–$x4$ arvoja.

Päätetään, että haluamme ennustaa muuttujan $x4$ arvon ajanhetkellä $t = 0$ käyttäen hyväksi sekä muuttujan itsensä aikaisempia arvoja $x4(-1)$, $x4(-2)$ sekä muiden muuttujien $x1(-1)$, $x2(-1)$ ja $x3(-1)$ arvoja ajanhetkellä $t = -1$. Opetuksessa käytettävät data-vektorit

$\mathbf{x}_i$ ja $\mathbf{y}_i$ muodostetaan liu'uttamalla aikaikkunaa taaksepäin annetussa datassa:

$$\mathbf{x}_1 = \begin{bmatrix} x1(-1) \\ x2(-1) \\ x3(-1) \\ x4(-1) \\ x4(-2) \end{bmatrix}, \quad \mathbf{y}_1 = \begin{bmatrix} x4(0) \end{bmatrix}; \qquad \mathbf{x}_2 = \begin{bmatrix} x1(-2) \\ x2(-2) \\ x3(-2) \\ x4(-2) \\ x4(-3) \end{bmatrix}, \quad \mathbf{y}_2 = \begin{bmatrix} x4(-1) \end{bmatrix}. \tag{4.34}$$

Siispä yleisessä muodossa asetetaan

$$\mathbf{x}_t = \begin{bmatrix} x1(-t) \\ x2(-t) \\ x3(-t) \\ x4(-t) \\ x4(-(t+1)) \end{bmatrix}, \quad \mathbf{y}_t = \begin{bmatrix} x4(-t+1) \end{bmatrix} \qquad t = 1, 2, \ldots \tag{4.35}$$

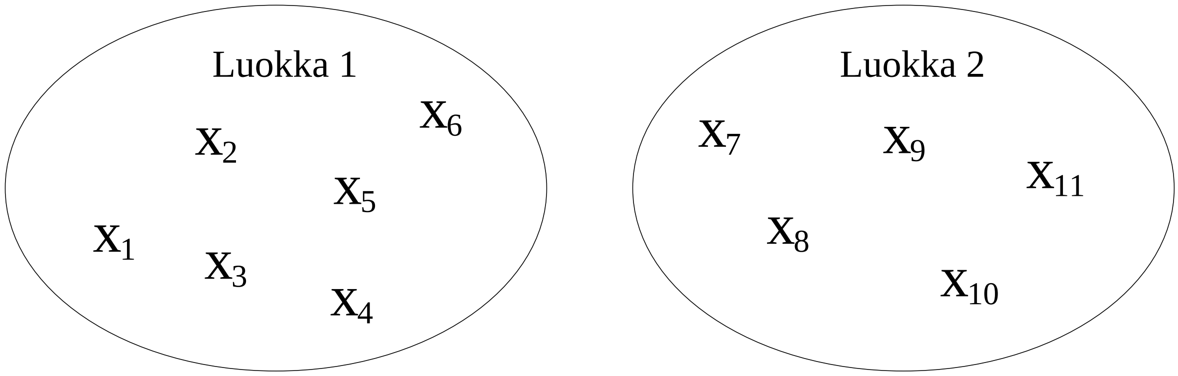
niin pitkään kuin $x4(-(t+1))$ on olemassa. Käytännössä homma luonnollisesti toteutetaan siten, että opetusaineiston muodostus -silmukkaa juoksutetaan datan alusta sopivaan indeksiin t asti. Jättämällä annetusta datasta $\{\mathbf{x}_i\}$ jokin osa (yleensä loppuosa) opetuksessa huomioimatta voidaan saadun ennustimen hyvyyttä testata opetustehtävän ratkaisemisen jälkeen tämän ylimääräisen ns. *testijoukon* avulla. Tämä systeemi yleistyy luonnollisesti myös muihin MLP-sovelluksiin.

Missä tällaista tekniikkaa voisi hyödyntää? Konkreettisena esimerkkinä voisi olla vaikkapa jonkun osakkeen (osakkeiden) pörssikurssin ennustaminen pörssistä kerättyjen tietojen (esim. valittujen osakkeiden ja optioiden kurssien kehitys nykyhetkeen saakka) avulla Toisena sovelluksena voitaisiin ajatella sään ennustamista jollakin paikkakunnalla kerättyjen lämpötila, ilmanpaine, tuulen suunta yms. mittausten avulla. Rahan tekoa voisi tietenkin yrittää myös ennustamalla erilaisten urheilutapahtumien tuloksia vanhojen tulosten avulla. Lisää mahdollisia esimerkkejä löytyy esim. viime vuonna vastaavalla kurssilla tehdyistä harjoitustöistä.

Perusmuodossaan MLP-verkko suorittaa ensin lineaarisen muunnoksen annetulle datavektorille $\mathbf{x}_i$. Esimerkiksi aikasarjojen ennustamisessa saattaisi joskus olla hyödyllistä muodostaa jo valmiiksi jonkinlainen epälineaarinen muunnos annetulle datalle. Epälineaarisuus saadaan helpoimmin aikaan soveltamalla jotain sopivaa (kanta)funktiota annettuun dataan ja muodostamalla näin uusia "kenttiä" käytettyihin datavektoreihin $\mathbf{x}_i$. Tällöin puhutaan *korkeamman asteen prosessointiyksiköistä* eli *sigma-pi* -yksiköistä. Edellä käsittelemämme esimerkin tapauksessa voitaisiin siis kaavassa (4.35) esiintyvän määrityksen sijaan käyttääkin muotoa

$$\mathbf{x}_t = \begin{bmatrix} x1(-t) \\ x1(-t)^2 \\ x2(-t) \\ x2(-t)^2 \\ x1(-t) * x2(-t) \\ \vdots \end{bmatrix} \qquad t = 1, 2, \ldots \tag{4.36}$$

Luonnollisesti ongelmana tässä kuten yleensä muissakin monimutkaisemmissa MLP-verkko -sovelluksissa on se, että verkon sisältämien tuntemattomien painojen (matriisien $\mathbf{W}^1$ ja $\mathbf{W}^2$ komponenttien) määrä kasvaa nopeasti niin suureksi, että verkon opetusalgoritmin vaatima suoritusaika räjähtää käsiin (ainakin ilman erikoistemppuja).



Kuva 4.5: Kaksi luokkaa ja niihin liittyvät datavektorit.

4.3.2 Luokittelu

MLP-verkkoja käytetään usein erilaisten epälineaaristen *luokittimien* muodostamiseen. Peruslähtökohtana on tällöin kokoelma datavektoreita $\{\mathbf{x}_i\}$, $\mathbf{x}_i \in \mathbf{R}^{n_0}$ kaikille i, joiden tiedetään kuuluvan eri luokkiin c_k, $k = 1, \ldots, K$, missä K on luokkien kokonaismäärä. Esimerkiksi pullonpalautusautomaatin yhteydessä luokat olisivat erityyppisiä pulloja, joiden panttisummat vaihtelevat. Yhden datavektorin $\mathbf{x}_i$ kentät (eli vektorin komponentit $x_{i,1}, x_{i,2}, \ldots, x_{i,n_0}$) voisivat kuvata esimerkiksi pullojoukosta eri korkeudelta mitattujen poikkileikkauksien pituuksia käytetyissä mittayksiköissä.

Luokittelutehtävän saattamiseksi MLP-verkon ymmärtämään muotoon tarvitaan tapa, jolla muodostetaan kuhunkin datavektoriin $\mathbf{x}_i$ liittyvä haluttu ulostulo $\mathbf{y}_i$ siten, että se vastaa havainnon tunnettua luokkaa c_k. Helpoin tapa luokan koodaamiseksi on käyttää luonnollisia kantavektoreita $\mathbf{e}_k \in \mathbf{R}^K$. Siis, jos $\mathbf{x}_i$ kuuluu luokkaan c_k, valitaan vektoriksi $\mathbf{y}_i$ kantavektori

$$\mathbf{y}_i^T = \mathbf{e}_k^T = [0\ \ldots\ \overset{k's}{1}\ \ldots\ 0]^T.$$

Kuvan 4.5 tapauksessa opetusaineiston halutuiksi ulostuloiksi valittaisiin tällöin

$$\mathbf{y}_i = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad i = 1, \ldots, 6; \qquad \mathbf{y}_i = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad i = 7, \ldots, 11. \tag{4.37}$$

Käytettyä valintaa (koodausta) tukee ainakin se tosiseikka (joka yleistyy myös useampien luokkien tapaukseen), että näin valitut luokkavektorit $\mathbf{y}_1$ ja $\mathbf{y}_2$ ovat *lineaarisesti riippumattomia* eli $\mathbf{y}_1^T \mathbf{y}_2 = 0$.

Opetettua MLP-verkkoa käytettäisiin varsinaisessa luokittelussa siten, että annettu datavektori $\mathbf{x}$ (pullonpalautusautomaattiin syötetystä uudesta pullosta mitatut poikkileikkaukset) sijoitettaisiin siihen luokkaan, joka vastaa suurinta arvoa verkon ulostulona antamasta muunnosvektorista $\mathbf{o} = \mathcal{N}(\mathbf{x})$. Siis, yleisessä tapauksessa valittaisiin

$$\mathbf{x} \in c_k, \quad \text{missä } k = \arg\max_j \mathbf{o}_j.$$

Käytetyssä kahden luokan esimerkissä tämä tarkoittaisi sitä (nyt siis $\mathbf{o}$ on $\mathbf{R}^2$:n vektori komponentein o_1 ja o_2), että luokaksi valittaisiin 'Luokka 1', jos $o_1 > o_2$, ja vastaavasti 'Luokka 2', jos $o_2 > o_1$. Tästä nähdäänkin suoraan, että ns. *luokkaraja* (jossa luokittelija ei osaa erottaa luokkia toisistaan) kyseisten kahden luokan välillä sijaitsee joukossa $\{\mathbf{x} \in \mathbf{R}^{n_0} : o_1 = o_2\}$.

Huomatetaan vielä lopuksi, että tämän havainnon pohjalta luokittelijan voi tehdä *varovaisemmaksi* valitsemalla sopivan kynnysarvon $0 < \theta \leq \frac{1}{2}$ ja asettamalla luokaksi

$$\begin{array}{ll} \text{'Luokka 1'}, & \text{jos } o_1 > o_2 + \theta, \\ \text{'Luokka 2'}, & \text{jos } o_2 > o_1 + \theta, \\ \text{'Luokittelematon'}, & \text{muussa tapauksessa.} \end{array}$$

Luokittelun yhteydessä kannattaa muistaa lemmassa 2 saatu tulos, jonka mukaan MLP-verkko konfiguroituu tilaan, joka vastaa haluttujen ulostulojen keskiarvoa. Siispä erityisesti "MLP-luokittelijan" muodostama luokkaraja vetäytyy aina sen luokan dataa kohti, jota on enemmän. Datan määrästä riippumattoman luokittelijan muodostamiseksi täytyy alkuperäistä kustannusta (4.14) *skaalata* sopivasti eri luokkia edustavien opetusvektoreiden $\{\mathbf{x}_i^k, \mathbf{y}_i^k\}$ (luokan ilmaisevana indeksinä käytettiin tässä siis k:ta) lukumäärien avulla (katso [TK]).

4.3.3 Tiedon kompressointi MLP-verkon avulla

MLP-verkon avulla voidaan myös pyrkiä pudottamaan annettujen datavektoreiden $\{\mathbf{x}_i\}$ dimensiota siten, että prosessissa hukkuu mahdollisimman vähän informaatiota. Tarkastellaan aluksi lineaarista tekniikkaa, ns. *pääkomponenttianalyysiä* (engl. PCA = Principal Component Analysis), joka liittyy läheisesti lineaarisen Perceptron-verkon käyttöön (muista **Askel 2** luvussa 4.1).

Pääkomponenttianalyysi[2]

Tavoitteena on muuntaa datavektorit $\{\mathbf{x}_i\}$, $\mathbf{x}_i \in \mathbf{R}^n$, uusiksi vektoreiksi $\{\mathbf{z}_i\}$, $\mathbf{z}_i \in \mathbf{R}^m$, siten, että toisaalta $m < n$ ja toisaalta esitys on mahdollisimman hyva eli $\mathbf{x}_i \sim \mathbf{z}_i$ "sopivassa mielessä". Kuten johdantokappaleessa opittiin, jokainen $\mathbf{R}^n$:n vektori $\mathbf{x}$ voidaan esittää joidenkin muiden (tässä vaiheessa kiinnittämättömien) ortonormaalien kantavektorien $(\mathbf{u}_1, \ldots, \mathbf{u}_n)$ avulla muodossa

$$\mathbf{x} = \sum_{k=1}^{n} z_k \mathbf{u}_k, \qquad \text{missä } z_k = \mathbf{u}_k^T \mathbf{x}. \tag{4.38}$$

Kaava $z_k = \mathbf{u}_k^T \mathbf{x}$ kertoimen z_k määräämiseksi saadaan, kun identiteetti (4.38) kerrotaan molemmilta puolilta k:nnella kantavektorilla $\mathbf{u}_k^T$ ja muistetaan lisäksi monisteen alkupuolella annettu määritelmä vektorijoukon $\{\mathbf{u}_i\}$ ortonormaalisuudelle. Muistetaan vielä se jo aiemmin mainittu seikka, että todellisuudessa kaava (4.38) suorittaa pelkästään käytetyn koordinaattisysteemin rotaation.

Luonnollinen tapa redusoida alkuperäisen vektorin $\mathbf{x}$ dimensiota on korvata se vektorilla $\mathbf{z}$, joka sisältää esityksessä (4.38) esiintyvät ne reaaliluvut z_k, $k = 1, \ldots, m$, joita vastaavat esityksen tarkkuuden kannalta *merkittävimmät* kantavektorit $\mathbf{u}_k$. Siis, tarkastellaan uutta vektoria

$$\tilde{\mathbf{x}} = \sum_{k=1}^{m} z_k \mathbf{u}_k + \sum_{k=m+1}^{n} b_k \mathbf{u}_k, \tag{4.39}$$

[2] TÄMÄKIN OSA ON TENTIN KANNALTA YLIMÄÄRÄISTÄ ASIAA!!!

missä jälkimmäinen termi $\sum_{k=m+1}^{n} b_k \mathbf{u}_k$ siis sisältää virheen, joka alkuperäisen vektorin $\mathbf{x}$ redusoitu esitystapa $\sum_{k=1}^{m} z_k \mathbf{u}_k$ sisältää. Luonnollisesti tehdyn virheen suuruus saadaan kaavojen (4.38) ja (4.39) avulla lausuttua muodossa

$$\mathbf{x} - \tilde{\mathbf{x}} = \sum_{k=m+1}^{n} (z_k - b_k)\, \mathbf{u}_k. \tag{4.40}$$

Kuten MLP-verkon opetuksenkin yhteydessä, vielä tässä vaiheessa tuntematonta vektorijoukkoa $\{\mathbf{u}_k\}$ haetaan siten, että koko annetun datajoukon $\{\mathbf{x}_i\}$ yli laskettu keskimääräinen virhe $\mathbf{x}_i - \tilde{\mathbf{x}}_i$ olisi mahdollisimman pieni. Siispä vektoreiden $\mathbf{u}_k$ määräämiseksi meidän tulee minimoida kustannusta

$$\widehat{\mathcal{J}}(\{\mathbf{u}_k\}) = \frac{1}{2}\sum_{i=1}^{N} \|\mathbf{x}_i - \tilde{\mathbf{x}}_i\|^2 = \frac{1}{2}\sum_{i=1}^{N} (\mathbf{x}_i - \tilde{\mathbf{x}}_i)^T (\mathbf{x}_i - \tilde{\mathbf{x}}_i) = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=m+1}^{n} (z_{i,k} - b_k)^2, \tag{4.41}$$

missä jälleen käytettiin vektoreiden $\{\mathbf{u}_k\}$ ortonormaalisuutta hyväksi. Derivoimalla (4.41) komponentin $\mathbf{b}_k$ suhteen ja asettamalla derivaatta nollaksi saadaan aikaiseksi lauseke

$$b_k = \frac{1}{N}\sum_{i=1}^{N} z_{i,k} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{u}_k^T \mathbf{x}_i = \mathbf{u}_k^T \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i = \mathbf{u}_k^T \bar{\mathbf{x}}, \tag{4.42}$$

missä $\bar{\mathbf{x}}$ on siis vektoreiden $\{\mathbf{x}_i\}$ *odotusarvo* (keskiarvo) $\frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$. Sijoittamalla esitykset (4.38) ($z_{i,k}$:lle) ja (4.42) (b_k:lle) kustannukseen (4.41), voidaan se kirjoittaa muodossa (käytetään kahdelle vektorille $\mathbf{u}$ ja $\mathbf{v}$ pätevää yleistä tulosta $(\mathbf{u}^T \mathbf{v})^2 = (\mathbf{u}^T \mathbf{v})(\mathbf{u}^T \mathbf{v}) = (\mathbf{u}^T \mathbf{v})(\mathbf{v}^T \mathbf{u}) = \mathbf{u}^T (\mathbf{v}\mathbf{v}^T)\, \mathbf{u}$)

$$\widehat{\mathcal{J}}(\{\mathbf{u}_k\}) = \frac{1}{2}\sum_{k=m+1}^{n}\sum_{i=1}^{N} \left(\mathbf{u}_k^T (\mathbf{x}_i - \bar{\mathbf{x}})\right)^2 = \frac{1}{2}\sum_{k=m+1}^{n} \mathbf{u}_k^T \Sigma\, \mathbf{u}_k, \tag{4.43}$$

missä matriisi Σ on ns. *kovarianssimatriisi*

$$\Sigma = \sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T. \tag{4.44}$$

Nyt homma onkin selvä. Kaavassa (4.43) saatiin minimoitava kustannus esitettyä kvadraattisessa muodossa kovarianssimatriisin Σ avulla. Kun jälleen muistellaan kappaleessa 1 esitettyjä asioita matriisiin liittyvälle kvadraattiselle muodolle, saadaan johtopäätös, jonka mukaan vektorit $\mathbf{u}_k$ ovat (symmetrisen ja positiivisesti ainakin semidefiniittisen) matriisiin Σ ominaisvektoreita. Koska nämä olivat erityisesti ortonormaaleja, voidaan (4.43) saattaa lopulta muotoon

$$\widehat{\mathcal{J}}(\{\mathbf{u}_k\}) = \frac{1}{2}\sum_{k=m+1}^{n} \lambda_k, \tag{4.45}$$

missä λ_k on vastaavasti kovarianssimatriisin Σ k's ominaisarvo. Tämä tarkoittaa sitä, että kaavassa (4.39) määriteltyyn redusoituun (kompressoituun) esitykseen $\sum_{k=1}^{m} z_k \mathbf{u}_k$ valitaan m-kappaletta ominaisvektoreita, jotka vastaavat m:ää matriisin Σ suurinta ominaisarvoa.

SUMMA SUMMARUM: *datan lineaarinen muunnos, joka säilyttää suurimman määrän informaatiota, on datavektoreiden projisointi kovarianssimatriisin suurimpia ominaisarvoja vastaaville ominaisvektoreille. Prosessissa hukkuneen informaation määrää kuvaa jäännöstermi kaavassa* (4.45).

Huomautus 2. Pääkomponenttianalyysillä saadaan siis määrättyä lineaarinen muunnos, joka redusoi datan dimensiota hukkaamalla mahdollisimman vähän (valitun määrän) informaatiota. Kun muistetaan, että varsinaisen MLP-verkon ensimmäisessä kerroksessa tehdään myöskin ensin lineaarinen muunnos, huomataan, että monissa tapauksissa pääkomponenttianalyysiä voidaan/kannattaa soveltaa annetun opetusdatan esikäsittelyssä ennen varsinaista MLP-verkon opetusta.

MLP-verkon avulla suoritettu kompressointi

Tarkastellaan lopuksi varsinaisesti sitä, miten annetun datan dimensiota voitaisiin pudottaa MLP-verkon avulla. Periaate on sama kuin pääkomponenttianalyysissäkin eli haluamme esittää annetut datavektorit pienemmällä määrällä muuttujia siten, että prosessissa hukkuu mahdollisimman vähän informaatiota. Yleisessä tapauksessa tämä vastaa sitä, että data aluksi *koodataan* jollakin menetelmällä $\mathcal{K}$ pienemmälle määrälle muuttujia ja käänteisoperaationa on redusoidun datan *dekoodaus* menetelmällä $\mathcal{D}$, joka palauttaa alkuperäisiä datavektoreita "mahdollisimman hyvin". Formaalisti esitettynä pyritään siis siihen, että

$$\mathbf{x}_i \sim \mathcal{D}(\mathcal{K}(\mathbf{x}_i)) \tag{4.46}$$

kaikille datavektoreille $\mathbf{x}_i$, $i = 1, \ldots, N$.

MLP-verkon tapauksessa peruslähtökohtana on käyttää kooderina $\mathcal{K}$ ja dekooderina $\mathcal{D}$ alku- ja loppuosaa monikerroksisesta Perceptron-verkosta siten, että tavoite (4.46) toteutuu mahdollisimman hyvin. Aluksi opetusdatan halutuiksi ulostuloiksi $\mathbf{y}_i$ valitaan siten *täsmälleen samat* datavektorit $\mathbf{x}_i$ kaavan (4.46) mukaisesti. Varsinaiseksi MLP-verkoksi voidaan sitten valita esimerkiksi *nelikerroksinen* rakenne (hirvittääkö?)

$$\mathbf{W}^4\, \widehat{\mathcal{F}}^2(\mathbf{W}^3\, \widehat{\mathbf{W}}^2\, \widehat{\mathcal{F}}^1(\mathbf{W}^1\, \hat{\mathbf{x}})), \tag{4.47}$$

missä $\mathbf{W}^1 \in \mathbf{R}^{n_1 \times n_0+1}$, $\mathbf{W}^2 \in \mathbf{R}^{n_2 \times n_1+1}$, $\mathbf{W}^3 \in \mathbf{R}^{n_3 \times n_2+1}$ ja $\mathbf{W}^4 \in \mathbf{R}^{n_4 \times n_3+1}$. Tällöin painomatriisin $\mathbf{W}^2$ rivien lukumäärä n_2 sisältää redusoiduille datavektoreille asetetun dimension, jolle siis valitaan $n_2 < n_0$ (n_0:lla merkittiin alkuperäisten datavektoreiden $\mathbf{x}_i$ dimensiota). Tekemässämme valinnassa kaavaan (4.46) sisältyvä kooderi on siis $\mathcal{K}(\mathbf{x}) = \mathbf{W}^2\, \widehat{\mathcal{F}}^1(\mathbf{W}^1\, \hat{\mathbf{x}})$ ja dekooderi $\mathcal{D}(\mathbf{z}) = \mathbf{W}^4\, \widehat{\mathcal{F}}^2(\mathbf{W}^3\, \hat{\mathbf{z}})$. Kuten aikaisemmin pääkomponenttianalyysin yhteydessä huomattiin, saadaan prosessissa tehdyn virheen suuruus suoraan kustannusfunktion $\mathcal{J}\left(\{\mathbf{W}^l\}_{l=1}^4\right)$ arvosta, joka on laskettu lopullisille (opetusalgoritmin antamille) painomatriiseille ($\mathbf{W}^1$, $\mathbf{W}^2$, $\mathbf{W}^3$, $\mathbf{W}^4$).

Kuten ylläolevasta tavasta huomataan, perusongelmana MLP-verkon käytölle datan kompressointiin on jälleen neljän painomatriisin sisältämien, alunperin tuntemattomien, komponenttien suuri määrä. Siksi myöskin tässä yhteydessä *tehokkaiden* (= vähän muistia ja CPU-aikaa vaativien) opetusalgoritmien kehittäminen ja soveltaminen on välttämätöntä.