Lappeenrannan teknillinen yliopisto
Teknillistaloudellinen tiedekunta
Tietotekniikan koulutusohjelma

Kandidaatintyö

Epälineaaristen regressiomenetelmien vertaaminen robottikäden otemallinnuksessa

Työn ohjaajana toimii DI Janne Laaksonen
Kandidaatintyön aihe on hyväksytty 21.9.2011

Niko Reunanen
niko.reunanen@lut.fi
2011

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Teknillistaloudellinen tiedekunta
Tietotekniikan koulutusohjelma

Niko Reunanen

Epälineaaristen regressiomenetelmien vertaaminen robottikäden otemallinnuksessa

2011

44 sivua, 19 kuvaa, 12 taulukkoa, 2 algoritmia, 1 liite

Tarkastaja: DI Janne Laaksonen

Hakusanat: MLP, RBFN, SVR, RVR, robottikäsi, otemallinnus

Tässä työssä verrattiin monikerrosperseptronin, radiaalikantafunktioverkon, tukivektoriregression ja relevanssivektoriregression soveltuvuutta robottikäden otemallinnukseen. Menetelmille ohjelmoitiin koeympäristö Matlabiin, jossa mallit koestettiin kolmiulotteisella kappaledatalla. Koejärjestely sisälsi kaksi vaihetta. Kokeiden ensimmäisessä vaiheessa menetelmille haettiin sopivat parametrit ja toisessa vaiheessa menetelmät koestettiin. Kokeilla kerättiin dataa menetelmien keskinäiseen vertailuun. Vertailussa huomioitiin laskentanopeus, koulutusaika ja tarkkuus. Tukivektoriregressio löydettiin potentiaaliseksi vaihtoehdoksi mallintamiseen. Tukivektoriregression koetuloksia analysoitiin muita menetelmiä enemmän hyvien koetulosten takia.

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology Management
Department of Information Technology

Niko Reunanen

Comparing the usage of nonlinear regression methods in modeling the grasp of a robot hand

2011

44 pages, 19 figures, 12 tables, 2 algorithms, 1 appendice

Examiner: M.Sc. Janne Laaksonen

Keywords: MLP, RBFN, SVR, RVR, robot hand, grasp modeling

The thesis compared the suitability of multilayer perceptron, radial basis function network, support vector regression and relevance vector regression in modeling the grasp of a robot hand. Matlab was used to program a test environment for the methods, where the models were tested using the data of three dimensional objects. The test procedure consisted of two phases. First, suitable parameters were searched for the methods, and then the methods were tested. The tests collected data for method comparison. The compared attributes were the speed of calculation, training time and accuracy. Support vector regression was found as a potential method for the modeling. Test results of the support vector regression were analyzed more than the other methods, because the support vector regression gave the best test results.

Sisällysluettelo

1 JOHDANTO **3**

1.1 Tavoitteet ja rajaukset 4

1.2 Työn rakenne 5

2 ROBOTTIKÄDEN OTTEEN MALLINNUS **6**

3 MENETELMIEN ESITTELY **7**

3.1 Monikerrosperseptroni 8

3.2 Radiaalikantafunktioverkko 9

3.3 Tukivektoriregressio 11

3.4 Relevanssivektoriregressio 13

4 KOKEET **17**

4.1 Monikerrosperseptroni 19

4.1.1 Parametrien hakutulokset 19

4.1.2 Koetulokset 20

4.2 Radiaalikantafunktioverkko 20

4.2.1 Parametrien hakutulokset 21

4.2.2 Koetulokset 21

4.3 Tukivektoriregressio 22

4.3.1 Parametrien hakutulokset 22

4.3.2 Koetulokset 23

4.4 Relevanssivektoriregressio 24

4.4.1 Parametrien hakutulokset 24

4.4.2 Koetulokset 25

5 TULOSTEN VERTAILU **26**

5.1 Tukivektoriregression tulosten analysointi 27

6 YHTEENVETO **31**

LÄHTEET **32**

LIITE 1. Kokeiden Matlab - koodi **34**

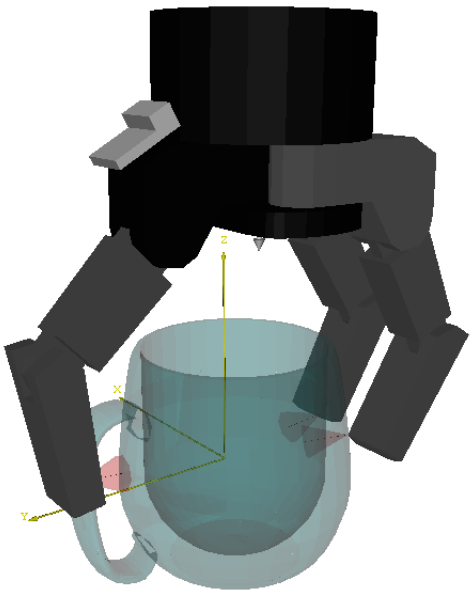
Symboliluettelo

MLP	Monikerrosperseptroni	Multilayer Perceptron
MSE	Virheen neliön keskiarvo	Mean Square Error
RBFN	Radiaalikantafunktioverkko	Radial Basis Function Network
RVR	Relevanssivektoriregressio	Relevance Vector Regression
SVM	Tukivektorikone	Support Vector Machine
SVR	Tukivektoriregressio	Support Vector Regression

$\mathcal{N}$	Normaalijakauma
α_i	Hyperparametri
$\boldsymbol{\alpha}$	Hyperparametrivektori
γ_s	Leveysparametri
γ_{reg}	Regularisoinnin suuruuskerroin
ϵ	Virhetaso
ϕ	Ydinfuktio
σ	Leveysparametri
σ^2	Varianssi
ξ	Virheen sallittavuus
b	Vakioarvo
c	Keskipiste
C	Kompromissiarvo
d	Opetusesimerkin ulostuloarvo
K	Ydinfunktio
N	Ulostulojen määrä
x	Syöte
y	Ulostulo
Y	Ulostulojen joukko
w	Painoarvo
$\boldsymbol{w}$	Painoarvovektori

1 JOHDANTO

Tässä työssä robottikäsi tietää tartuntaotteestaan kappaleeseen ainoastaan sormien nivelkulmat. Kappaleen sijainti, kulma ja otteen stabiilisuus pitää päätellä. Tässä työssä stabiilisuus tarkoittaa kappaleen pysyvyyttä otteessa robottikättä nostettaessa. Halutut tiedot voidaan määrittää tartuntasimulaatiolla, mutta prosessi on liian hidas ja epäkäytännöllinen. Työssä vertaillaan neljän eri epälineaarisen regressiomenetelmän soveltuvuutta otteen mallintamiseen. Robottikäden tartuntaotteessa (kuva 1) vaikuttaa kuusi vapausastetta: robotin ja kappaleen suhteellinen sijainti kolmiulotteisessa karteesisessa koordinaatistossa sekä suhteellinen orientaatio.

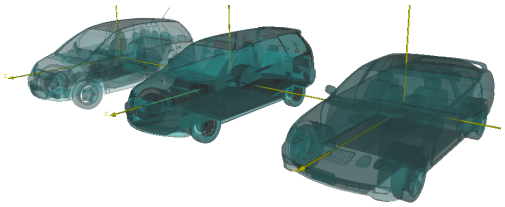


Kuva 1: Kolmisormisen robottikäden ote

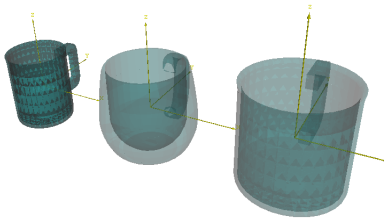
Kandidaatintyössä vapausastearvoista pyritään päättelemään kappalekohtaisesti sormien nivelkulmien arvot sekä otteen stabiilisuus. Ongelmaksi muodostuu mallien rakentaminen ja opettaminen halutulle toimintatarkkuudelle. Toimivaa mallia voidaan käyttää tulevaisuudessa esimerkiksi Monte Carlo - menetelmien kanssa päättelemään kappaleen todennäköinen sijainti ja otteen stabiilisuus.

1.1 Tavoitteet ja rajaukset

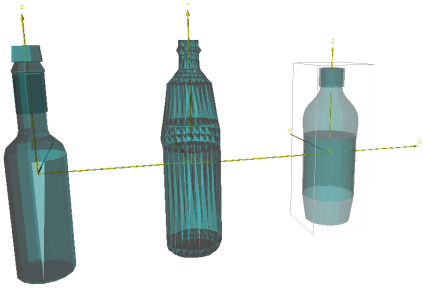
Työn tavoitteena on *vertailla mallien soveltuvuutta sormien nivelkulmien ja otteen stabiilisuuden laskemiseen*. Matemaattiset mallit ovat kaksi keinotekoista neuroverkkoa ja kaksi vektoriregressiota. Neuroverkot ovat monikerrosperseptroni (MLP) ja radiaalikantafunktioverkko (RFBN). Vektoriregressiot ovat tukivektoriregressio (SVR) ja relevanssivektoriregressio (RVR). Mallit toteutetaan ja ajetaan Matlab – ohjelmiston koodikirjastoilla. Mallit esitellään pintapuolisella teoreettisella tasolla. Matemaattisten mallien tuottamat tulokset validoidaan testisyötteillä. Työssä käytetään DI Janne Laaksosen simuloimalla keräämää dataa. Pitkäkestoisen datakeräyksen takia valmiin datan käyttö nopeuttaa kehitystyötä. Data sisältää kappaleita neljässä eri kategoriassa ja jokainen kategoria sisältää kolme erilaista kappaletta. Kategoriat ovat auto (kuva 2), muki (kuva 3), pullo (kuva 4) ja vasara (kuva 5). Jokainen kappale sisältää 9075 paria vapausastearvoja, sormien nivelkulmia ja stabiilisuusarvoja.



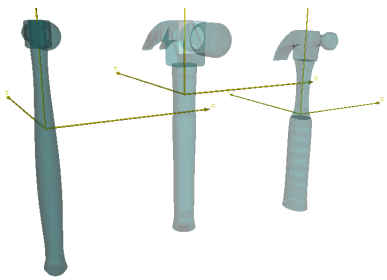
Kuva 2: Autot



Kuva 3: Mukit



Kuva 4: Pullot



Kuva 5: Vasarat

1.2 Työn rakenne

Työ rakentuu kuudesta luvusta. Toisessa luvussa kerrotaan teoriaa robottikäden otteesta ja esitellään kaksi aikaisempaa mallinnusratkaisua. Kolmannen luvun alussa esitetään käytettyjen matemaattisen menetelmien yleistä teoriaa. Menetelmäkohtaiset kuvaukset ja teoriat esitetään omissa aliluvuissa. Luvussa neljä esitellään työn koejärjestelyt ja koestetaan menetelmät. Ensiksi parametreille etsitään sopivat asetukset, jonka jälkeen ajetaan kokeet. Kokeista kirjataan tuloksia vertailun pohjaksi. Luvussa viisi verrataan syntyneitä tuloksia, pohditaan mallien käyttökelpoisuutta ja analysoidaan tukivektoriregression koetuloksia. Luvussa kuusi kirjoitetaan yhteenveto työstä ja tuloksista sekä tarkastetaan suunnitellun aikataulun toteutuminen.

2 ROBOTTIKÄDEN OTTEEN MALLINNUS

Robottikäsi vuorovaikuttaa ympäristön ja kappaleiden välillä sormillaan. Robottisormien nivelkulmien toiminta-alueella voidaan muodostaa mielivaltaisia nivelkulmayhdistelmiä. Robottikäsi voi tarttua kappaleeseen ihmiskäden tapaan monesta eri sijainnista ja kulmasta. Sormilla varustettu robottikäsi kykenee näppärään interaktioon ympäristön kanssa. Robottikäden näppäryys määritellään kykynä muuttaa kappaleen sijaintia ja orientaatiota. Näppäryys nostaa otemallinnuksen kompleksisuutta. Yksinkertaisessa otehaussa robottikäsi tarttuu simulaatiossa kappaleeseen monta kertaa ja yrittää kartoittaa hyviä otteita. Tapa ei ole kuitenkaan käytännöllinen suuren ajankulutuksen takia. Otemallinnukseen voidaan rakentaa matemaattinen malli ennustamaan otteen laatu vaihtuvissa olosuhteissa. Ideaalitapauksessa malli selviytyy otemallinnuksesta tarjoamalla hyvän otteen kappaleelle silloinkin, kun vaikuttavat voimat ei ole tiedossa. [Bic00, SK08]

Aikaisemmat tutkimustyöt sisältävät useita geometriaan perustuvia ratkaisuja stabiilisten otteiden löytämiseen. Vuonna 2008 tutkittiin tapaa, jossa kolmiulotteisista pisteistä koostuva kappale leikattiin muotonsa perusteella pienempiin laatikoihin. Tutkijat kehittivät algoritmin, jolla leikattujen laatikoiden tilavuus minimoidaan. Leikatut laatikot ovat arvioita kappaleen muodosta ja jakavat kappaleen tartuttaviin osiin. Menetelmän käyttöä perustellaan nopealla laskennallisuudella sekä mahdollisuudella simuloida osien otteen stabiilisuus. Otteiden stabiilisuus laskettiin GraspIt! – simulaatioympäristöllä. [HRK08]

Columbia Grasp Database - projektissa rakennettiin vuonna 2009 tietokanta erilaisista kappaleista ja niiden stabiileista otteista. Kappaleiden geometriat olivat tietokannan indeksejä stabiileille otteille. Tarttuminen on vahvasti riippuvainen robottikäden rakenteesta, joten stabiilit otteet etsittiin kahdelle eri robottikädelle. Tutkijat ehdottivat, että tarkasteltavalle kappaleelle etsitään tietokannasta geometrisesti samankaltainen kappale esimerkiksi k:n lähimmän naapurin menetelmällä. Tutkijoita motivoi idea, että uusien kappaleiden tarttumiseen käytetään tietokannasta olevien samankaltaisten kappaleiden esilaskettuja stabiileja otteita. [GCDA09]

3 MENETELMIEN ESITTELY

Luokitteluun ja regressioon käytettäviä matemaattisia menetelmiä ovat monikerrosperseptroni, radiaalikantafunktioverkko, tukivektoriregressio ja relevanssivektoriregressio. Menetelmät oppivat syötearvojen x ja ulostuloarvojen y välisen yhteyden (kaava 1). Regressiossa ulostulot ovat jatkuvia arvoja ja luokittelussa esimerkiksi binaariarvoja. Menetelmät approksimoivat mallinnettavassa ilmiössä esiintyvää funktiota. Oppimisprosessi käsitetään approksimointiin sopivan funktion ja parametrien päättelemisenä [Tip00].

$$\{x_n\}_{n=1}^{N} \rightarrow \text{malli} \rightarrow \{y_n\}_{n=1}^{N} \tag{1}$$

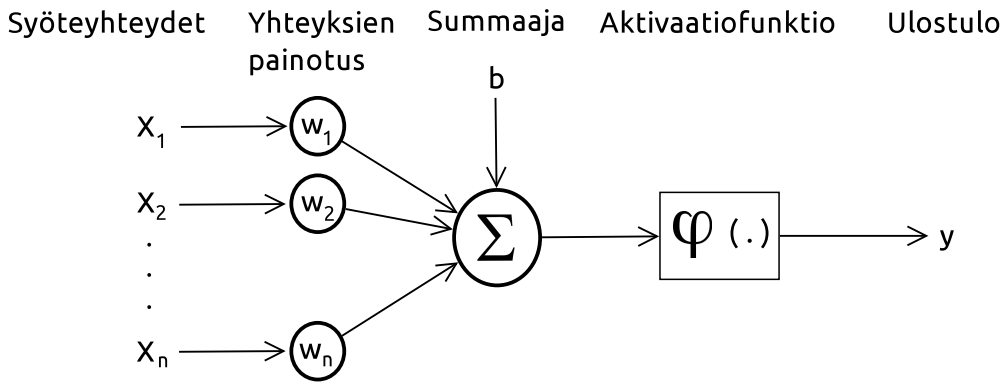
Sisään- ja ulostuloarvot ovat numeerisia arvoja, joten syötteet koodataan numeroarvoiksi. Mallit koulutetaan sisään- ja ulostulojen esimerkkipareilla. Opetusta esimerkeillä kutsutaan ohjatuksi oppimiseksi. Mallit yleistävät tietoa, jolloin tuntemattomilla syötearvoilla malli päättelee vastauksen oppimansa perusteella. Opetusdata saattaa kuitenkin sisältää kohinaa, joten malli ei saa ylisovittua dataan, koska ylisovittumisessa malli oppii opetusdatan tarkasti menettäen yleistämiskyvyn. Mallit määrittävät koulutukselleen optimointiongelman, kuten esimerkiksi funktion minimoinnin. Tyypillinen optimointiongelma on mallin laskeman virheen minimointi. Kaavassa 2 esitetään MSE - virheen minimointi, jossa Y on ulostulojen joukko, n on opetusesimerkin indeksi, d on opetusesimerkin ulostuloarvo, y on mallin laskema ulostuloarvo ja i ulostulon indeksi.

$$\begin{aligned} e_i(n) &= |y_i(n) - d_i(n)| \\ \min \ & \frac{1}{N} \sum_{i \in Y} e_i(n)^2 \end{aligned} \tag{2}$$

Tulevissa aliluvuissa esitellään työn menetelmät seuraavassa järjestyksessä: MLP, RBFN, SVR ja RVR. Esittely koostaa menetelmien oleellisen teorian tekstillä, kuvilla ja kaavoilla.

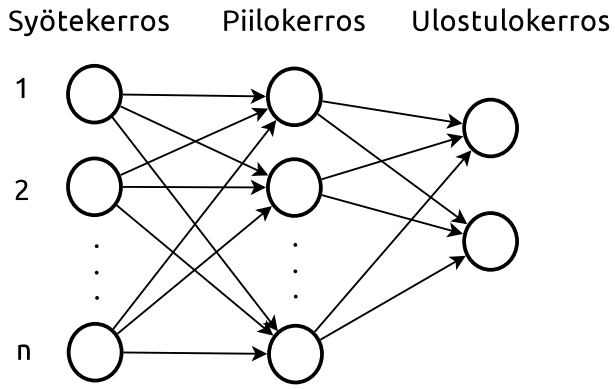
3.1 Monikerrosperseptroni

Keinotekoinen neuroni on MLP:n tärkein rakenneosa, joka pohjautuu Frank Rosenblattin Perceptroniin. Kuvassa 6 esitetään neuronin rakenne. Neuroni koostuu aktivaatiofunktiosta, summaajasta, vakioarvosta sekä syöte- ja ulostuloyhteyksistä. Jokainen yhteys on painotettu ja kuljettaa arvon. Summaaja laskee yhteen painotetut syöteyhteydet sekä vakioarvon. Aktivaatiofunktio saa syötteeksi summaajan luvun. Lopuksi neuroni kuljettaa aktivaatiofunktion arvon ulostuloyhteyksiin [Ros58]. Tieto ja oppiminen koodataan yhteyksien painoarvoiksi.



Kuva 6: Keinotekoisen neuronin rakenne

MLP rakentuu kerroksiin asetetuista keinotekoisista neuroneista. Kerrokset ovat syötekerros, yksi tai useampi piilokerros ja ulostulokerros (kuva 7). Jokaisessa kerroksessa on yksi tai useampi neuroni. Kokonaan kytketyssä MLP:ssa kerroksen jokainen neuroni on yhdistetty seuraavan kerroksen kaikkiin neuroneihin. Osittain kytketystä rakenteesta neuronien välisiä yhteyksiä puuttuu. Neuronit ovat laskentayksiköitä, ja laskentasignaali kulkee aina eteenpäin. MLP:n topologia määrittää kerrosten määrän, neuroneiden määrän kerroksissa sekä muodostettujen yhteyksien määrän. [Hay99, sivut 181–182]



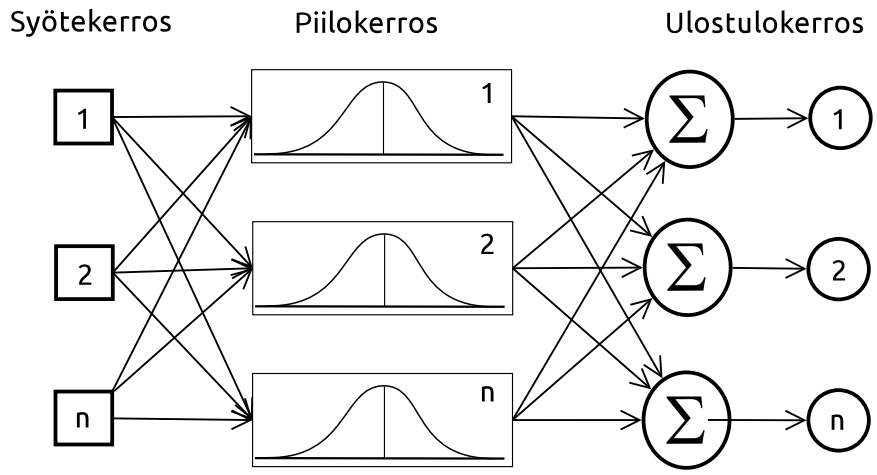
Kuva 7: MLP:n neuronikerrokset

Back-propagation on tunnettu algoritmi MLP:n kouluttamiseen [Hay99, sivu 178]. Algoritmi optimoi virhefunktion muuttamalla neuronien painoarvoja. Muutokset perustuvat gradienttioptimointiin. Gradienttioptimoinnin ongelmana ovat lokaalit minimit, jolloin back-propagation ei laske parasta ratkaisua [KS96, sivut 39–40]. Virhefunktio voidaan regularisoida rankaisemaan ulostulojen virheen lisäksi isoja painoarvoja (kaava 3) [Lam97]. Parametri γ_{reg} määrittää regularisoinnin suuruuden. Pienet painoarvot w_j parantavat esimerkiksi ekstrapolointia ja yleistyskykyä.

$$
\begin{aligned}
e_i(n) &= |y_i(n) - d_i(n)| \\
\min \quad & \frac{1}{N} \sum_{i \in Y} e_i(n)^2 + \gamma_{reg} \sum_j w_j^2
\end{aligned}
\tag{3}
$$

3.2 Radiaalikantafunktioverkko

Radiaalikantafunktioverkko optimoi sovituksen koulutusesimerkeille korkeammassa ulottuvuudessa. Coverin teoreeman mukaisesti monimutkainen ongelma muuttunee lineaarisesti separoituvaksi korkeammassa ulottuvuudessa, kunhan tila-avaruus ei ole tiheästi asutettu [Hay99, sivut 278–279]. RBFN koostuu syötekerroksesta, ulostulokerroksesta ja yhdestä tai useammasta piilokerroksesta (kuva 8). Piilokerros sisältää epälineaariset kantafunktiot.



Kuva 8: Radiaalikantafunktioverkon rakenne

MLP:sta poiketen RBFN:n ulostulot ovat aina lineaarisia. Kaavassa 4 esitetään RBFN:n käyttämä funktiomallinnuksen approksimaatio. Piilokerroksen aktivointifunktiot saavat syötteeksi syötevektorin ja neuronin keskuksen c välisen euklidisen etäisyyden. Aktivointifunktio on radiaalikantafunktiosta rakennettu ydinfunktio ϕ. Ydinfunktio on tavallisesti Gaussin funktio.

$$y(x) = \sum_{i=1}^{N} w_i \phi(|x - c_i|) \tag{4}$$

Suurempi etäisyys tarkoittaa neuronin pienempää vaikutusvaltaa ja aktivaatiota. Aktivointifunktiot eroavat keskenään toisistaan, koska esimerkiksi Gaussin funktio voi saada eroavia arvoja parametreiksi. Aktivointifunktion arvo painotetaan painoarvolla, jotka määritetään kouluttamalla RBFN. Opittu tieto koodataan RBFN:ssä painojen arvoiksi ja ydinfunktion parametrien arvoiksi. Ulostulokerroksessa piilokerroksesta saadut painotetut arvot summataan yhteen jokaiselle ulostulolle. Summattu arvo on ulostuloarvo.

Radiaalikantafunktioverkon koulutus on kaksivaiheinen. Opetusesimerkeistä päätellään sopivat ydinfunktioiden parametrit. Esimerkiksi Gaussin ydinfunktiossa etsitään sopiva keskus. Sitten etsitään sopivat painoarvot $\boldsymbol{w}$. Yleinen ratkaisu on minimoida mallin virhe muuttamalla painoarvoja (kaava 2).

3.3 Tukivektoriregressio

Tukivektoriregressio on tukivektorikoneen regressiolaajennus. SVR käsittelee epälineaarisen regression korkeammassa ulottuvuudessa Coverin teoreeman mukaisesti. Tavoitteena on löytää kaavaan 5 perustuva regressiofunktio, joka ennustaa arvoja opetuksen pohjalta parhaiten.

$$y(x) = w^T \phi(x) + b \tag{5}$$

Approksimoinnin ydinfunktiona voidaan käyttää esimerkiksi polynomista funktiota (kaava 6) tai radiaalikantafunktiota.

$$K(x, y) = (\gamma_s x^T y + b)^d \tag{6}$$

Tyypillinen radiaalikantafunktio on Gaussin funktio:

$$K(x, y) = \exp\left(-\gamma_s |x - y|^2\right) \tag{7}$$

SVR rakentaa harvan regressiomallin käyttämällä datapisteiden osajoukkoa. Valittuja datapisteitä kutsutaan tukivektoreiksi. Kaavan 5 regressiomalli $y(x)$ rakennetaan ratkaisemalla kaavan 8 optimointiongelma. C määrittää virheen minimoinnin ja mallin monimutkaisuuden kompromissin, ϵ regressoijan virheherkkyyden ja ξ_i virheen sallittavuutta regression eri datapisteissä. [Vap99, SL05]

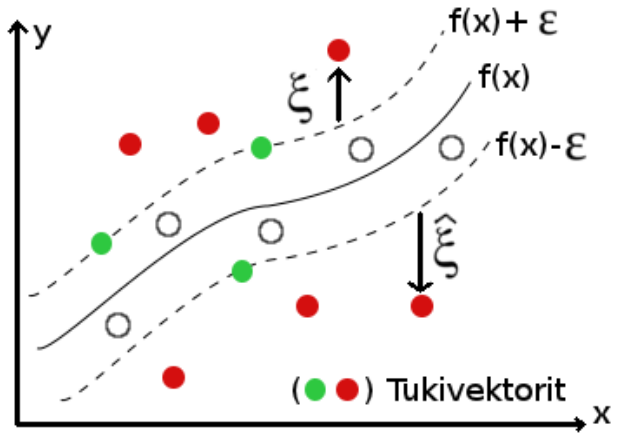
$$\begin{aligned} \min \quad & \frac{||\boldsymbol{w}||^2}{2} + C \sum_{i=0}^{l} \xi_i + C \sum_{i=0}^{l} \hat{\xi}_i \\ & \text{ehdot:} \\ & d_i - w^T \phi(x_i) - b \leq \epsilon + \xi_i \\ & w^T \phi(x_i) + b - d_i \leq \epsilon + \hat{\xi}_i \\ & \xi_i, \hat{\xi}_i \geq 0, i = 1, \ldots l \\ & \epsilon \geq 0 \end{aligned} \tag{8}$$

Kaavan 8 minimointi ratkaistaan Lagrangen menetelmällä ja ottamalla käyttöön Lagrangen kertoimet a_n ja $\hat{a}_n$. Uusien syötearvojen ennusteille saadaan kaavan 9 mukaisen mal-

lin sijoittamalla Lagrangen funktion osittaisderivaatta $\boldsymbol{w}$ suhteen kaavaan 5. [Bis06, sivut 326–343]

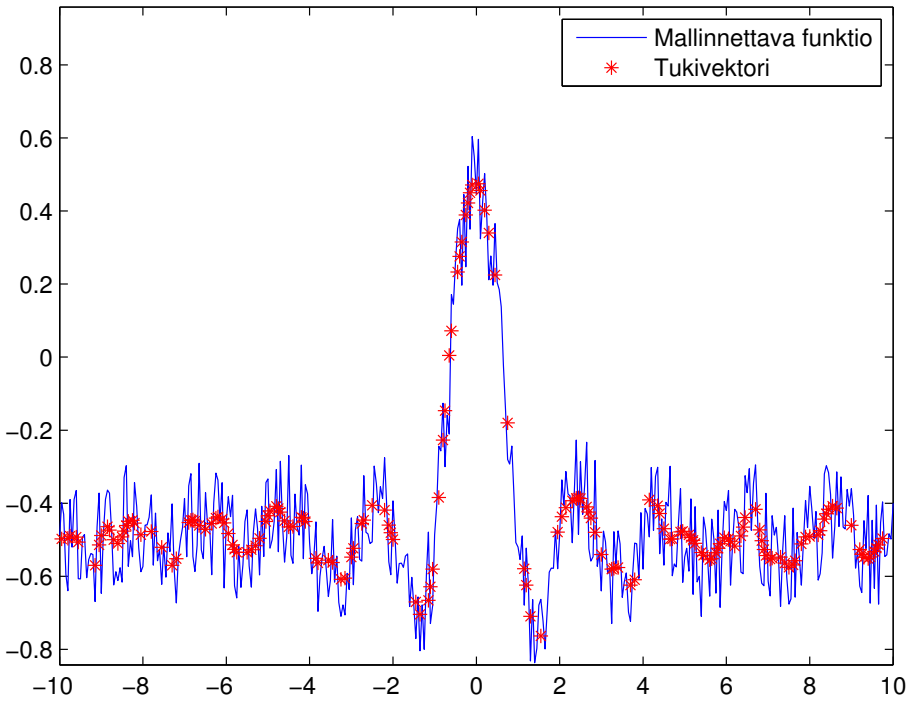
$$y(x) = \sum_{n=1}^{N} (a_n - \hat{a}_n) k(x, x_n) + b \tag{9}$$

Mallin ennusteet perustuvat piste-estimaatteihin eivätkä todennäköisyyteen. Regression laatu pohjautuu parametriarvoihin ja sopivan ydinfunktion valintaan. Sopivat parametriarvot haetaan ristikkohaulla, jossa kokeillaan parametriyhdistelmiä rajatulla alueella. Kuvassa 9 esitetään SVR, joka käyttää regressioon datan osajoukkoa eli tukivektoreita. SVR pyrkii kasvattamaan etäisyyden tukivektoreihin mahdollisimman suureksi ja jakamaan datapisteet mahdollisimman puhtaasti erilleen.



Kuva 9: SVR:n regressio ja tukivektorit

Kuvassa 10 esitetään normalisoidun sinc - funktion (kaava 10) mallinnus tukivektoriregressiolla ja käytetyt tukivektorit. Mallinnusdataan lisättiin satunnainen häiriösignaali. Tukivektoriregressio käytti tukivektoreina yhteensä 171 eri datapistettä.



Kuva 10: Tukivektorit häiriöllisen sinc - funktion mallinnuksessa

$$sinc(x) = \frac{sin(\pi x)}{\pi x} \tag{10}$$

3.4 Relevanssivektoriregressio

Vuonna 2000 Tipping siirsi SVM:n funktionaalisen idean bayesilaiseen päättelyyn. Mallin ennusteet perustuvat todennäköisyyteen. Kaavassa 11 on esitetty RVR:n regressiomalli. Ulostuloarvojen y_n oletetaan pohjautuvan oikeisiin arvoihin d_n häiriösignaalilla ϵ_n. Häiriösignaali on normaalijakautunut tuntemattomalla varianssilla. [Fle10]

$$\begin{aligned} d_n &= y(x_n) + \epsilon_n \\ y(x_n) &= w^T \phi(x_n) \\ \epsilon_n &\sim \mathcal{N}(0, \sigma^2) \\ p(d_n|x) &= \mathcal{N}(d_n|y(x_n), \sigma^2) \end{aligned} \tag{11}$$

RVR tavoittelee harvuutta eli käytettyjen ydinfunktioiden määrän minimointia. Mallista saadaan harva asettamalla usea painoarvo nollaan. Harvamallinen regressio ei ylisovitu opetusdataan herkästi, joten RVR:lla on hyvä yleistämiskyky. RVR käyttää vähemmän ydinfunktioita kuin SVR. RVR ei sisällä vapaita parametreja, joten ristikkohakua ei tarvita. Ainoastaan ydinfunktion leveysparametri γ_s etsitään käytettäessä Gaussin ydinfunktiota. Ylisovituksen ehkäisemiseksi painoarvokasvua rajoitetaan. Malli regularisoidaan painoarvojen esitietoa rajaavalla erillisellä priorijakaumalla. Vaihtoehto priorijakaumaksi on kaavan 12 normaalijakauma, missä hyperparametrivektori $\boldsymbol{\alpha}$ sisältää painoarvokohtaiset varianssiparametrit. Jokaisella painolla w_i on oma hyperparametri α_i, joka vaikuttaa priorin voimakkuuteen. [Tip01]

$$P(\boldsymbol{w}|\boldsymbol{\alpha}) \;=\; \prod_{i=1}^{M} \mathcal{N}(w_i|0, \alpha_i^{-1}) \qquad (12)$$
$$\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_M)^T$$

Usean hyperparametrin arvo lähestyy ääretöntä, jolloin vastaavat painoarvot asetetaan nollaksi. Seurauksena vastaavat ydinfunktiot poistuvat ja malli harvenee. Jäljelle jääneitä painoja kutsutaan relevanssivektoreiksi. Hyperparametrien arvot määritetään maksimoimalla kaavan 13 reunauskottavuus.

$$\ln p(t|\boldsymbol{\alpha}, \sigma^{-2}) \;=\; \frac{N}{2}\ln\sigma^{-2} - E(t) - \frac{1}{2}\ln|\sum| - \frac{N}{2}\ln 2\pi + \frac{1}{2}\sum_{i=1}^{M}\ln\alpha_i \qquad (13)$$

Reunauskottavuudesta (kaava 13) lasketaan osittaisderivaatat hyperparametrien suhteen ja muodostetaan yhtälöt $\frac{\partial p}{\partial \alpha_i} = 0$ ja $\frac{\partial p}{\partial \sigma^{-2}} = 0$. Yhtälöistä saadaan hyperparametreille kaavan 14 iteratiiviset estimointifunktiot reunauskottavuuden (kaava 13) maksimointiin.

Estimointifunktiossa $\mathbf{\Phi}$ on $N \times M$ matriisi, jonka rivi i esittää vektoria $\phi(x_i)$.

$$\begin{aligned} \alpha_i &= \frac{1 - a_i \sum_{ii}}{m_i^2} \\ \sigma^{-2} &= \frac{N - \sum i\gamma_i}{||\boldsymbol{t} - \mathbf{\Phi}\boldsymbol{m}||^2} \end{aligned} \tag{14}$$

missä

$$\begin{aligned} \boldsymbol{m} &= \sigma^{-2} \sum \mathbf{\Phi}^T \boldsymbol{t} \\ \sum &= (diag(\boldsymbol{\alpha}) + \sigma^{-2}\mathbf{\Phi}^T\mathbf{\Phi})^{-1} \end{aligned}$$

Mallin koulutuksen iteratiivinen prosessi on seuraava:

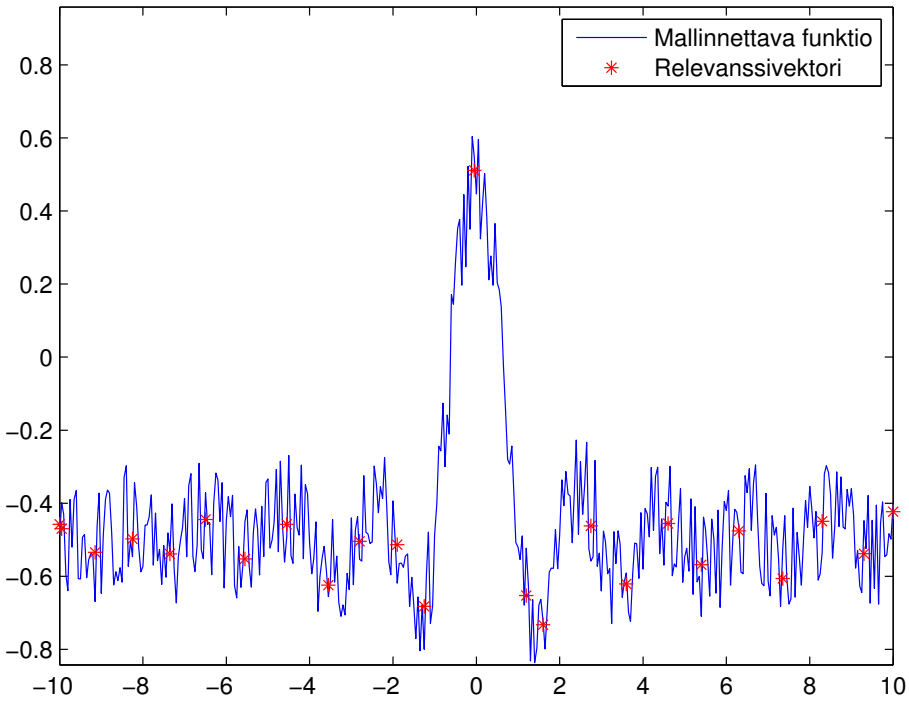
1. Valitaan alkuarvot hyperparametreille $\boldsymbol{\alpha}$ ja σ^{-2}.
2. Lasketaan $\boldsymbol{m}$ ja $\sum$ (kaava 14).
3. Päivitetään α_i ja σ^{-2} (kaava 14).
4. Poistetaan ääretöntä lähestyvät hyperparameterit.
5. Toistetaan vaiheet 2 – 4, kunnes hyperparametreille asetettu suppenemiskriteeri täyttyy.

Reunauskottavuuden maksimoivien hyperparametrien avulla voidaan arvioida ennustejakauma uusilla syötteillä (kaava 15), jolloin estimoitu ulostulo on ennustejakauman keskiarvo (kaava 16). [Fle10, Tip01]

$$P(t|x, \boldsymbol{\alpha}, \sigma^{-2}) = \mathcal{N}(\boldsymbol{m}^T\phi(x), \sigma^2(x)) \tag{15}$$

$$\boldsymbol{m}^T\phi(x) \tag{16}$$

Kuvassa 11 esitetään kuvan 10 häiriöllisen sinc - funktion mallinnus relevanssivektoriregressiolla ja käytetyt relevanssivektorit. Relevanssivektoriregressio käytti yhteensä 24 relevanssivektoria. Tukivektoriregressio käytti saman funktion mallintamiseen 171 tukivektoria, joten relevanssivektoriregression matemaattinen malli oli huomattavasti harvempi.



Kuva 11: Relevanssivektorit häiriöllisen sinc - funktion mallinnuksessa

4 KOKEET

Ennen kokeita menetelmille haetaan sopivat parametrit. Algoritmissa 1 esitetään menetelmän parametrihaku. Parametrihaussa MLP-, RBFN-, SVR- ja RVR-menetelmille etsitään 2-kertaisen ristiinvalidointivirheen minimoivat parametrit. Kaksikertaisessa ristiinvalidoinnissa malli koulutetaan puolella kappaledataa ja testataan lopulla (alg. 1, rivi 4,5). Jokainen kappale ristiinvalidoidaan, jolloin testidatan arvojen ja laskettujen arvojen välinen virheen neliön keskiarvo MSE lasketaan (alg. 1, rivi 5). Yksittäisen ristiinvalidointikierroksen aikana malli koulutetaan samoilla parametreilla. Virheiden MSE summalla perustellaan sopivien parametrien valinta (alg. 1, rivi 9). Parametrihaussa testattaville parametreille mitataan aikamääre, joka on mallin kaikille kappaleille käyttämien koulutus- ja laskenta-aikojen summa sekunneissa. Parametrihaun laskennallinen kompleksisuus kasvaa suureksi. Testausarvoille määritetään ala- ja ylärajat, jotta tulokset saadaan laskettua mielekkäässä ajassa. Ristiinvalidoinnin dataosat rakennetaan satunnaisesti valituilla datariveillä. Parametrihausta kirjataan ylös käytetty aika, parametrit ja MSE.

Algoritmi 1 Menetelmän parametrihaku

1: **while** testattavia parametreja jäljella **do**
2: **for** $k = 1 \rightarrow 12$ **do**
3: aseta testiparametrit malliin
4: kouluta malli puolella määrällä kappaleen k dataa
5: testaa malli lopulla dataa ja laske MSE_k
6: **end for**
7: laske $\sum_{k=1}^{12} MSE_k$
8: **end while**
9: **return** pienimmän $\sum_{k=1}^{12} MSE_k$ tuottamat parametrit

Algoritmissa 2 esitetään koerakenne. Kokeiden virhearvoissa eritellään sormien nivelkulmien ja stabiilisuuden MSE toisistaan. Jokaisen kappaleen data jaetaan kahteen osaan k_1 ja k_2 (alg. 2, rivi 3), jolloin k_1 sisältää 4538 datariviä ja k_2 4537 datariviä. Ensimmäinen osa sisältää indeksiltään parilliset datarivit ja toinen parittomat, jolloin datan spatiaalinen jakauma vastaa koko dataa. Kumpikin osa testataan kaikilla malleilla 10-kertaisella ristiinvalidoinnilla. Vuorotellen 90% (alg. 2, rivi 6) yksi osa datasta kouluttaa ja 10% testaa mallin (alg. 2, rivi 7). Kymmenestä ristiinvalidointitestauksesta lasketaan keskimääräinen

laskenta-aika $\overline{LA}$ sekä virhe sormien nivelkulmille ja stabiilisuudelle (alg. 2, rivit 9,10). Kokeissa laskenta-aika LA tarkoittaa 453 tuloksen laskemiseen kuluvaa aikaa sekunneissa. Sormien nivelkulmavirheet MSE_{sor} ja stabiilisuusvirheet MSE_{stab} ovat oikeiden arvojen ja laskettujen arvojen välinen MSE - virhearvo.

Algoritmi 2 Menetelmän koe

```
1: aseta haetut parametrit malliin
2: for k = 1 → 12 do
3:    jaa kappaleen k data kahteen osaan k_1 ja k_2
4:    for j = 1 → 2 do
5:       for i = 1 → 10 do
6:          kouluta malli 9/10 määrällä kappaleen k_j dataa
7:          testaa malli 1/10 määrällä dataa ja laske MSE_kij sekä LA
8:       end for
9:       laske sormien nivelkulmien ja otteen stabiilisuuden virhekeskiarvot MSE_kj
10:      laske keskimääräinen laskenta-aika LA
11:   end for
12: end for
```

Koetuloksista taulukoidaan dataosien k_1 ja k_2 keskimääräiset sormien nivelkulmavirheet $\overline{MSE_{sor1}}$ ja $\overline{MSE_{sor2}}$, stabiilisuusvirheet $\overline{MSE_{stab1}}$ ja $\overline{MSE_{stab2}}$ sekä keskimääräiset laskenta-ajat $\overline{LA_1}$ ja $\overline{LA_2}$. Vektoriregressioiden toteutuksissa SVR ja RVR kykenevät vain yhteen ulostuloarvoon, joten parametrihaussa ja kokeissa jokaiselle ulostulolle koulutetaan oma vektoriregressio. Kappaleiden syötedata normalisoidaan välille $[-1, 1]$ ja ulostulodata välille $[0, 1]$. Nivelkulma-arvojen normalisoimaton vaihteluväli on $0 - 2.5133$. Stabiilisuusarvojen normalisoimattomat vaihteluvälit esitetään taulukossa 1. Parametrihaussa ja kokeissa ilmoitetut MSE-arvot ovat normalisoituja. Matlabilla MSE-arvot lasketaan “mse” - funktiolla. Kokeet ajetaan tietokoneella, jossa on 2Gt keskusmuistia ja 2.7GHz kellotaajuuksinen tuplaydinprosessori.

Taulukko 1: Stabiilisuusarvojen normalisoimattomat vaihteluvälit

Kappale	Vaihteluväli
Auto 1	$0 - 1$
Auto 2	$0 - 1$
Auto 3	$0 - 1$
Muki 1	$0 - 0.946$
Muki 2	$0 - 0.653$
Muki 3	$0 - 0.638$
Pullo 1	$0 - 0.836$
Pullo 2	$0 - 1$
Pullo 3	$0 - 1$
Vasara 1	$0 - 0.302$
Vasara 2	$0 - 0.92$
Vasara 3	$0 - 1$

4.1 Monikerrosperseptroni

Monikerrosperseptroni luodaan Matlabin funktiolla "feedforwardnet" ja ajetaan funktiolla "sim". Kokeessa MLP:n topologia sisältää kaksi piilokerrosta. Koulutuksen lopetusvirhetasoksi valitaan esitiedon ja aiempien kokeilujen perusteella 0.0001 ja koulutuskierrosten maksimimääräksi 1000. Parametrihaussa sopiva piiloneuronimäärä haetaan iteratiivisesti. Hakuiteraatiossa ensimmäisen piilokerroksen neuronimääräksi valitaan n_1 ja toisen $n_2 = \frac{n_1}{5}$. Vaihtoehdoista $n_1 = 10, 15, 20, ..., 95$ valitaan pienivirheisin topologia. Regularisoinnin suuruuskertoimeksi valitaan esitiedolla $\gamma_{reg} = 0.75$. Aluksi MLP:n painoarvot ovat satunnaisia, joten uudelleenkoulutus antaa hieman eroavia tuloksia. Satunnaisuus huomioidaan hakemalla parametrit kaksi kertaa ja laskemalla tulosten keskiarvo.

4.1.1 Parametrien hakutulokset

Taulukossa 2 esitetään parametrihaun tulokset. Paras virhe $\sum MSE = 0.3578$ saavutettiin arvoilla $n_1 = 95$ ja $n_2 = 19$. Laskenta-aika kasvoi ja tuloksien virhesumma pieneni suhteessa piiloneuronien määrään.

Taulukko 2: MLP Parametrihakujen tuloskeskiarvot

n_1	n_2	$\sum MSE$	aika (s)	n_1	n_2	$\sum MSE$	aika (s)
10	2	1.1536	47.8692s	55	11	0.4435	836.468s
15	3	1.1067	76.639s	60	12	0.408	1194.49s
20	4	1.0679	105.7091s	65	13	0.416	1424.155s
25	5	0.9594	133.751s	70	14	0.3938	1805.8s
30	6	0.8426	204.857s	75	15	0.3878	2539.165s
35	7	0.6474	369.3185s	80	16	0.3886	2780.695s
40	8	0.5885	437.6645s	85	17	0.3842	3942.565s
45	9	0.4651	672.2075s	90	18	0.3772	4539.37s
50	10	0.4459	693.005s	95	19	0.3578	5303.26s

4.1.2 Koetulokset

Kokeeseen valittiin piilokerrosten neuronimääriksi $n_1 = 95$ ja $n_2 = 19$. Taulukossa 3 esitetään koetulokset. Tuloksista nähdään että MLP:n laskenta-aika oli hyvin pieni.

Taulukko 3: MLP Koetulokset

Kappale	$\overline{MSE_{sor1}}$	$\overline{MSE_{sor2}}$	$\overline{MSE_{stab1}}$	$\overline{MSE_{stab2}}$	$\overline{LA_1}$ (s)	$\overline{LA_2}$ (s)
Auto 1	0.0261	0.0264	0.0087	0.0083	0.1679s	0.1622s
Auto 2	0.0245	0.0236	0.0085	0.0096	0.0768s	0.0922s
Auto 3	0.0223	0.0245	0.0092	0.0095	0.0625s	0.0754s
Muki 1	0.0402	0.0405	0.0036	0.0051	0.0632s	0.1015s
Muki 2	0.0335	0.035	0.0083	0.0098	0.0637s	0.0815s
Muki 3	0.0357	0.031	0.008	0.0067	0.0752s	0.0675s
Pullo 1	0.0359	0.039	0.0096	0.0086	0.1191s	0.0954s
Pullo 2	0.034	0.0329	0.0061	0.0075	0.0692s	0.0693s
Pullo 3	0.0319	0.0342	0.0081	0.0078	0.062s	0.0656s
Vasara 1	0.0668	0.0644	0.0013	0.002	0.0625s	0.0648s
Vasara 2	0.0625	0.0633	0.0038	0.0029	0.0635s	0.0656s
Vasara 3	0.0583	0.0602	0.0018	0.0016	0.0691s	0.0643s

4.2 Radiaalikantafunktioverkko

Kolmikerroksinen radiaalikantafunktioverkko luodaan Matlabin omalla “newrb”-funktiolla. Ydinfunktion leveysparametri σ haetaan kokeilemalla $\sigma = 0.1, 0.2, ..., 1.5$. Matlab lisää radiaalikantafunktioverkon koulutusiteraatiossa piilokerrokseen neuroneita yksitellen. Tavoitteena on löytää ongelmaan sopiva pienin piiloneuronimäärä. Lopetusvirhetasoksi valitaan 0.001 ja piiloneuronien maksimimääräksi 200, jotta tulokset lasketaan mielekkäässä ajassa. Koulutus päättyy kun piiloneuronien maksimimäärä tai lopetusvirhetaso saavutetaan.

4.2.1 Parametrien hakutulokset

Taulukossa 4 esitetään parametrihaun tulokset. Paras virhe $\sum MSE = 0.2706$ saavutettiin arvolla $\sigma = 0.7$. Laskenta-aika pysyi lähes samana parametrin σ eri arvoilla. Piiloker-roksessa oli käytetty 200 piiloneuronia jokaisessa tuloksessa.

Taulukko 4: RBFN Parametrihaku

σ	$\sum MSE$	piiloneuronit	aika (s)	σ	$\sum MSE$	piiloneuronit	aika (s)
0.1	1.1131	200	3607.46s	0.9	0.2725	200	3323.71s
0.2	0.4973	200	3154.22s	1	0.2817	200	3264.36s
0.3	0.3282	200	3138.42s	1.1	0.2796	200	3219.55s
0.4	0.293	200	3136.67s	1.2	0.2818	200	3328.6s
0.5	0.2817	200	3137.18s	1.3	0.2831	200	3300.76s
0.6	0.2768	200	3137.55s	1.4	0.2913	200	3340.84s
0.7	0.2706	200	3121.62s	1.5	0.3048	200	3156.61s
0.8	0.2752	200	3241.42s				

4.2.2 Koetulokset

Taulukossa 5 esitetään koetulokset. Kokeissa ydinfunktion leveysparametri oli $\sigma = 0.7$.

Taulukko 5: RBFN Koetulokset

Kappale	$\overline{MSE_{sor1}}$	$\overline{MSE_{sor2}}$	$\overline{MSE_{stab1}}$	$\overline{MSE_{stab2}}$	$\overline{LA_1}$ (s)	$\overline{LA_2}$ (s)
Auto 1	0.0205	0.0204	0.0065	0.0053	0.1553s	0.1216s
Auto 2	0.0195	0.0198	0.0058	0.0062	0.1218s	0.1217s
Auto 3	0.019	0.0194	0.0066	0.0066	0.1218s	0.1213s
Muki 1	0.0274	0.0271	0.0028	0.0038	0.1218s	0.1219s
Muki 2	0.0241	0.0246	0.0069	0.0083	0.1217s	0.1214s
Muki 3	0.0254	0.025	0.0067	0.005	0.1222s	0.1229s
Pullo 1	0.0257	0.0261	0.0077	0.0066	0.121s	0.1223s
Pullo 2	0.0235	0.0241	0.005	0.006	0.1216s	0.1215s
Pullo 3	0.0249	0.0255	0.0061	0.0064	0.1227s	0.1242s
Vasara 1	0.053	0.0498	0.0017	0.0023	0.1229s	0.122s
Vasara 2	0.0468	0.0471	0.0034	0.0026	0.1262s	0.1207s
Vasara 3	0.0426	0.0432	0.0018	0.0014	0.1244s	0.122s

4.3 Tukivektoriregressio

Tukivektoriregression toteuttaa Chih-Chung Changin ja Chih-Jen Lin LibSVM – koodikirjasto [HCL03]. LibmSVM saadaan käyttöön kääntämällä C++ - kielinen lähdekoodi Matlabin mexglx - formaattiin. Funktiolla “svmtrain” koulutetaan SVR ja “svmpredict” lasketaan tulokset. Ydinfunktioksi valitaan kaavan 17 gaussin funktio.

$$K(x, y) = \exp\left(-\gamma_s |x - y|^2\right) \tag{17}$$

Parametrit etsitään ristikkohaulla, jossa kokeillaan parametrien C, γ_s ja ϵ arvoyhdistelmät rajatulla alueella. Taulukossa 6 esitetään kokeiltavat arvot. Parametriyhdistelmiä on yhteensä 1100 kappaletta. Ensimmäisessä haussa kokeillaan C ja γ_s arvoja välillä $0, 1$ ja toisessa haussa isompia arvoja.

Taulukko 6: SVR Ristikkohaku

	Ensimmäinen haku	Toinen haku
C	0.01,0.11,0.21,...,0.91	1,2,...,10
γ_s	0.05,0.1,0.15,...,1	0.5,1,...,5
ϵ	0.005,0.01,0.015,0.02	0.0005,0.02525,0.05
Yhdistelmiä	800 kpl	300 kpl

4.3.1 Parametrien hakutulokset

Taulukossa 7 esitetään parametrihaun 30 parasta tulosta. Paras virhe $\sum MSE = 0.2373$ saavutettiin parametrien asetuksilla $C = 3$, $\gamma_s = 5$ ja $\epsilon = 0.02525$. Mallin laskenta vei huomattavasti vähemmän aikaa kuin keinotekoisilla neuroverkoilla. Kaikki 30 parasta tulosta löytyivät toisesta hausta.

Taulukko 7: SVR Parametrihaun 30 parasta tulosta

C	γ_s	ϵ	$\sum MSE$	aika (s)	C	γ_s	ϵ	$\sum MSE$	aika (s)
3	5	0.02525	0.2373	141.13s	7	4	0.0005	0.2414	788.71s
4	4.5	0.0005	0.2379	490.24s	2	4.5	0.05	0.2415	101.42s
3	5	0.0005	0.2386	381.25s	5	4.5	0.0005	0.2415	578.42s
2	5	0.0005	0.2387	287.47s	3	4	0.02525	0.2417	145.92s
2	5	0.02525	0.2391	117.50s	4	4.5	0.02525	0.2417	165.96s
4	5	0.02525	0.2398	163.03s	8	4	0.0005	0.2417	868.51s
3	4.5	0.02525	0.2399	144.46s	3	4.5	0.05	0.2417	117.40s
5	4	0.0005	0.24	592.23s	2	4	0.02525	0.2418	122.43s
2	4.5	0.0005	0.24	295.99s	3	4	0.0005	0.2418	406.94s
4	5	0.0005	0.2401	473.38s	2	4.5	0.02525	0.2418	120.65s
5	5	0.0005	0.2401	556.73s	6	4.5	0.02525	0.2419	209.62s
1	5	0.0005	0.2404	182.47s	6	4	0.0005	0.2419	686.69s
7	3.5	0.0005	0.2405	830.88s	5	4	0.02525	0.2419	191.21s
4	4	0.0005	0.2407	495.34s	3	5	0.05	0.2419	114.11s
7	4.5	0.0005	0.2414	759.81s	3	4.5	0.0005	0.242	394.07s

4.3.2 Koetulokset

Taulukossa 8 esitetään koetulokset. Kokeeseen valittiin parametrit $C = 3$, $\gamma_s = 5$ ja $\epsilon = 0.02525$. Tulokset osoittivat että laskenta-aika pysyi pienenä ja yleistyskyky oli hyvä.

Taulukko 8: SVR Koetulokset

Kappale	$\overline{MSE_{sor1}}$	$\overline{MSE_{sor2}}$	$\overline{MSE_{stab1}}$	$\overline{MSE_{stab2}}$	$\overline{LA_1}$ (s)	$\overline{LA_2}$ (s)
Auto 1	0.0179	0.0191	0.0069	0.0056	0.1925s	0.1893s
Auto 2	0.0179	0.0183	0.0068	0.0071	0.1937s	0.1867s
Auto 3	0.0179	0.0177	0.0063	0.007	0.1982s	0.1882s
Muki 1	0.021	0.0214	0.003	0.0037	0.2179s	0.2083s
Muki 2	0.0202	0.0203	0.0072	0.009	0.2198s	0.2158s
Muki 3	0.0228	0.0225	0.0074	0.0056	0.2322s	0.2138s
Pullo 1	0.0246	0.0251	0.0083	0.008	0.2439s	0.2327s
Pullo 2	0.0227	0.0228	0.0057	0.0069	0.2265s	0.2233s
Pullo 3	0.0224	0.0224	0.0065	0.0068	0.2246s	0.2217s
Vasara 1	0.0493	0.0501	0.0013	0.0026	0.2712s	0.2581s
Vasara 2	0.0458	0.0471	0.0035	0.0031	0.2783s	0.262s
Vasara 3	0.0407	0.0405	0.0022	0.0018	0.2564s	0.2507s

Tukivektoriregressio kulutti vähän muistia ja oli erittäin nopea kouluttaa, joten SVR ajettiin myös koko kappaledatalla puolittamatta dataa. Taulukossa 9 esitetään täyden kappa-

ledatan 10-kertaisen ristiinvalidoinnin koetulokset. Taulukoitu aikamääre $\overline{LA}$ tarkoittaa 907 tuloksen keskimääräistä laskenta-aikaa sekunneissa.

Taulukko 9: SVR Koetulokset täydellä datalla

Kappale	$\overline{MSE_{sor}}$	$\overline{MSE_{stab}}$	$\overline{LA}$ (s)
Auto 1	0.0147	0.0046	0.5971s
Auto 2	0.0144	0.005	0.5896s
Auto 3	0.0142	0.0049	0.5926s
Muki 1	0.0173	0.0026	0.6686s
Muki 2	0.0166	0.0063	0.6995s
Muki 3	0.0187	0.0045	0.7052s
Pullo 1	0.0195	0.0054	0.7618s
Pullo 2	0.0177	0.0042	0.7398s
Pullo 3	0.0177	0.0044	0.6884s
Vasara 1	0.039	0.0017	0.8696s
Vasara 2	0.037	0.0024	0.8735s
Vasara 3	0.0317	0.0017	0.8471s

4.4 Relevanssivektoriregressio

Relevanssivektoriregression Matlabille toteuttaa Michael Tippinging SparseBayes 2.0 -koodikirjasto [TF03]. Kokeessa SparseBayes ajetaan “Probabilistic Modeling Toolkit for Matlab, version 3” - työkalukokoelman avulla [Mur10]. Funktiolla “rvmFit” koulutetaan ja funktiolla “rvmPredict” ajetaan RVR. Ydinfunktioksi valitaan kaavan 17 Gaussin funktio. Leveysparametri haetaan kokeilemalla arvoja $\gamma_s = 0.25, 0.5, ..., 3$.

4.4.1 Parametrien hakutulokset

Taulukossa 10 esitetään parametrihaun tulokset. Paras virhe $\sum MSE = 0.2673$ saavutettiin leveysparametrin arvolla $\gamma_s = 3$. Menetelmän laskenta-aika kasvoi voimakkaasti parametrin γ_s suuremmilla arvoilla.

Taulukko 10: RVR Parametrihaku

γ_s	$\sum MSE$	aika (s)	γ_s	$\sum MSE$	aika (s)
0.25	0.4972	503.862s	1.75	0.2959	3699.55s
0.5	0.4288	837.682s	2	0.2892	4248.68s
0.75	0.3824	1435.02s	2.25	0.2806	5117.72s
1	0.3511	1462.52s	2.5	0.2709	6326.26s
1.25	0.3256	2268.58s	2.75	0.268	7007.14s
1.5	0.3096	2727.13s	3	0.2673	8200.82s

4.4.2 Koetulokset

Taulukossa 11 esitetään koetulokset. Kokeeseen valittiin leveysparametrin arvoksi $\gamma_s = 3$. RVR:n laskenta-aika ja muistinkulutus olivat kaikkien kokeiden suurimmat.

Taulukko 11: RVR Koetulokset

Kappale	$\overline{MSE_{sor1}}$	$\overline{MSE_{sor2}}$	$\overline{MSE_{stab1}}$	$\overline{MSE_{stab2}}$	$\overline{LA_1}$ (s)	$\overline{LA_2}$ (s)
Auto 1	0.0195	0.02	0.0069	0.0058	0.6947s	0.7067s
Auto 2	0.019	0.0191	0.0068	0.0073	0.6953s	0.7367s
Auto 3	0.0185	0.0185	0.0072	0.0076	0.7351s	0.6929s
Muki 1	0.0243	0.0243	0.0029	0.0039	0.6775s	0.6829s
Muki 2	0.0216	0.022	0.0072	0.0083	0.7908s	0.6933s
Muki 3	0.0234	0.0228	0.0067	0.0052	0.6937s	0.6805s
Pullo 1	0.0276	0.0282	0.0081	0.0072	0.6879s	0.6605s
Pullo 2	0.0252	0.0253	0.0053	0.0065	0.6706s	0.6741s
Pullo 3	0.0254	0.0253	0.0068	0.0069	0.6627s	0.7147s
Vasara 1	0.0525	0.0526	0.0013	0.0023	0.6975s	0.6592s
Vasara 2	0.0485	0.0483	0.0034	0.0027	0.6876s	0.682s
Vasara 3	0.0452	0.0456	0.0017	0.0015	0.6965s	0.6927s

5 TULOSTEN VERTAILU

Tukivektoriregressio oli laskenta-aikoja tarkasteltaessa nopein menetelmä. Varsinaisissa kokeissa neuroverkkojen ja RVR:n koulutukseen kului aikaa useita tunteja, kun SVR puolestaan kouluttautui minuuteissa. Tukivektoriregressio oli myös ainoa menetelmä, jonka vähän muistia kuluttava koulutus mahdollisti kappaledatan täyden käytön. Muilla menetelmillä täysimääräisen datan koe ei onnistunut, vaan data jaettiin kahteen osaan. Radiaalikantafunktioverkon hidas koulutusaika pysyi lähes vakiona riippumatta leveysparametrin arvosta. Muiden menetelmien parametrivalinnoilla oli suora vaikutus koulutusaikaan.

Nopein menetelmä tulosten laskemiseen oli MLP ja hitain RVR, jonka huonoimmassa tapauksessa ristiinvalidoinnin testidatan keskimääräinen laskenta-aika oli $0.791s$. Täydellä kappaledatalla SVR laski suunnilleen samalla nopeudella kuin RVR. Radiaalikantafunktioverkko laski tulokset lähes vakionopeudella $0.12s$, kun puolestaan monikerrosperseptronin laskenta-ajat vaihtelivat välillä $0.062s$–$0.168s$. Kaikkien menetelmien laskenta-aika oli alle yhden sekunnin. Tukivektoriregressio laski tulokset alle sekunnissa myös täydellä kappaledatalla.

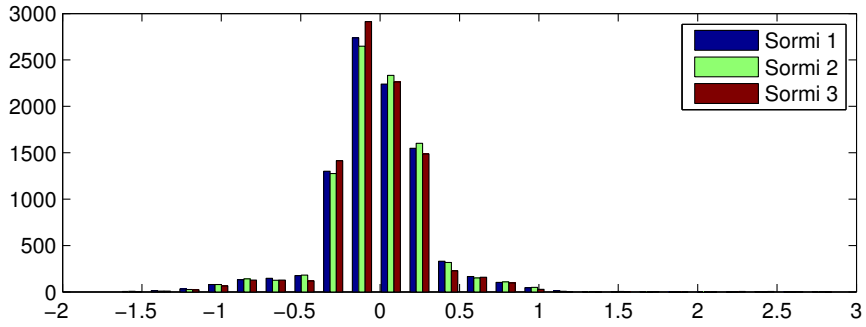
Kappalekategorioista vasarat tarjosivat suurimman haasteen menetelmille sormien nivelkulma-arvojen mallinnuksessa. Toisaalta stabiilisuuden mallinnus onnistui parhaiten juuri vasaroilla. Monikerrosperseptronin nivelkulmien laskenta-arvovirheiden vaihteluväli oli 0.0223–0.668, RBFN:n 0.019–0.053, SVR:n 0.0177–0.0501 ja RVR:n 0.0185–0.0526. Monikerrosperseptronin stabiilisuuksien laskenta-arvovirheiden vaihteluväli oli 0.0013–0.0098, RBFN:n 0.0014–0.0083, SVR:n 0.0013–0.009 ja RVR:n 0.0013–0.0083. Vaihteluväleistä nähdään, että SVR laski pienimmän virheen ja yleisti tietoa parhaiten. Työssä käytetyistä menetelmistä *SVR sopi parhaiten sormien nivelkulmien ja stabiilisuuden mallintamiseen*. Muut menetelmät eivät yleistäneet kokeissa yhtä hyvin, olivat hitaampia ja kuluttivat enemmän muistia. Tukivektoriregression huono puoli on ristikkohaku parametreille, jossa pitää hakea kolmen parametrin yhdistelmäarvot. Huonoiten mallinnustyöhön sopi MLP, joka laski suurimman yleistysvirheen. Taulukossa 12 esitetään menetelmien vertailun sanallinen yhteenveto.

Taulukko 12: Menetelmien vertailun yhteenveto

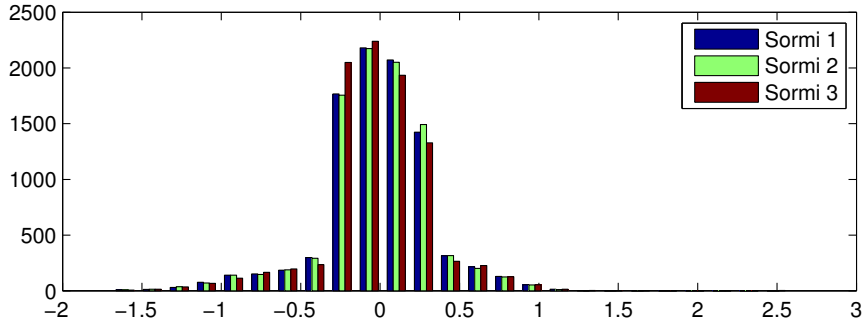
	MLP	R BFN	SVR	RVR
Koulutusaika	Hidas	Hidas	Nopea	Hidas
Laskenta-aika	Nopea	Nopea	Nopea	Nopea
Muistinkulutus	Suuri	Suuri	Pieni	Suuri
Yleistyskyky	Huono	Hyvä	Hyvä	Hyvä

5.1 Tukivektoriregression tulosten analysointi

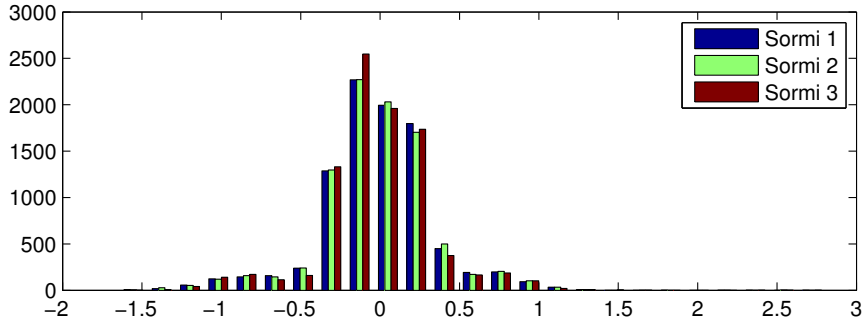
Tukivektoriregression koulutusaika oli menetelmistä lyhin ja muistikompleksisuus ei kasvanut liian isoksi edes koko datan käytöllä. Hyvien koetulosten takia SVR:n käyttöä tutkittiin muita menetelmiä pidemmälle. Tukivektoriregressiolla laskettiin kappalekategorioiden ensimmäisten kappaleiden sormien nivelkulmien ja otteen stabiilisuuden puhtaat virhearvot. Puhdas virhearvo tarkoittaa mallin ulostuloarvon ja oikean arvon normalisoimatonta erotusta. Virhearvot visualisoidaan histogrammeilla. Auton (kuva 12), mukin (kuva 13), pullon (kuva 14) ja vasaran (kuva 15) sormien nivelkulmien virhehistogrammiarvot vaikuttavat normaalijakautuneilta.



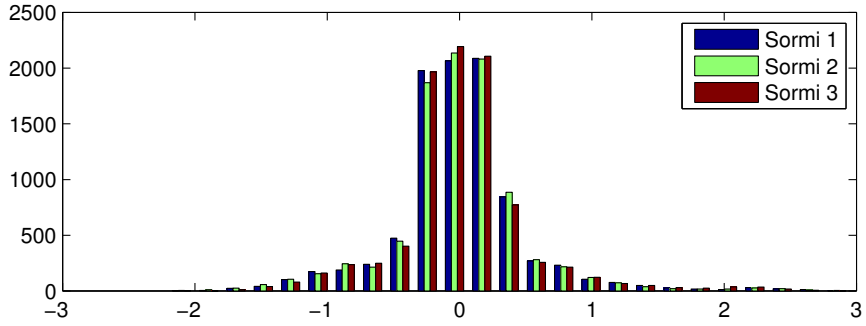
Kuva 12: Kappaleen “Auto 1” sormien nivelkulmavirheiden histogrammi



Kuva 13: Kappaleen “Muki 1” sormien nivelkulmavirheiden histogrammi



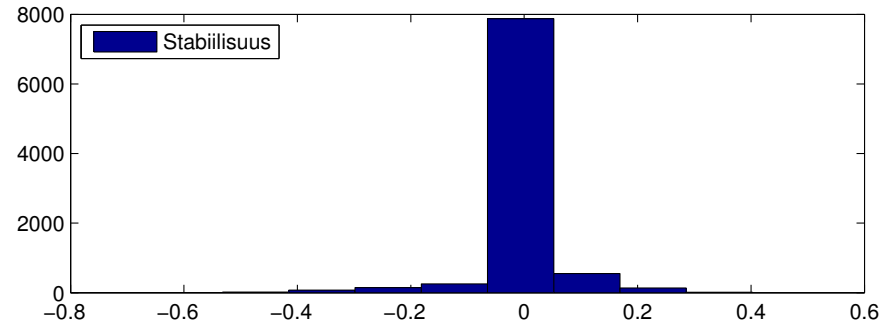
Kuva 14: Kappaleen “Pullo 1” sormien nivelkulmavirheiden histogrammi



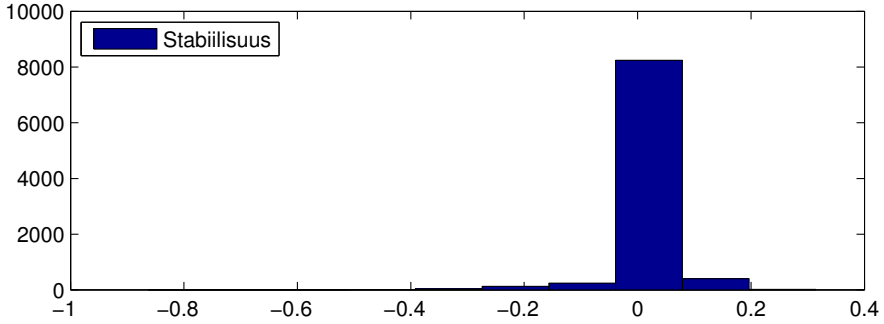
Kuva 15: Kappaleen “Vasara 1” sormien nivelkulmavirheiden histogrammi

Auton (kuva 16), mukin (kuva 17), pullon (kuva 18) ja vasaran (kuva 19) stabiilisuusvirheiden histogrammeissa oli eniten arvoja luokassa 0. Tällöin SVR laski otteen stabiilisuu-

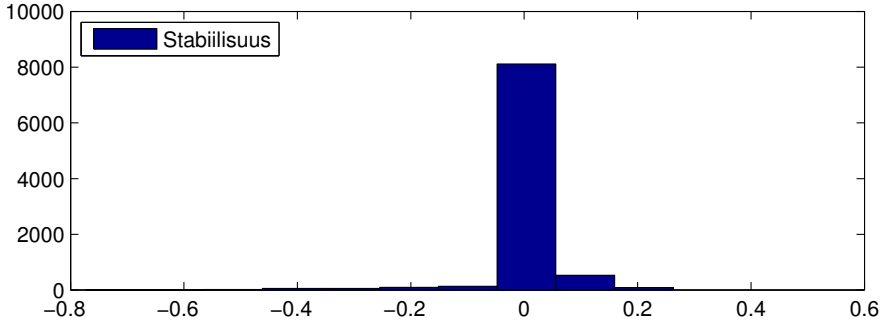
den oikein, olipa kyseessä stabiili ote tai ei. Eniten virhelaskentoja oli luokan 0 oikealla puolella, jolloin SVR laski otteelle oikeata isomman stabiilisuuden.



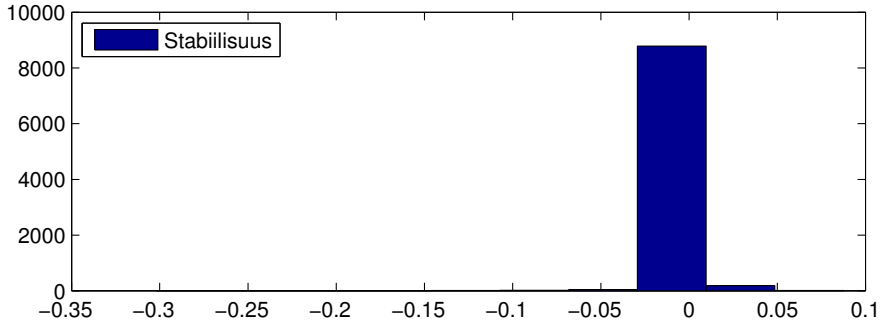
Kuva 16: Kappaleen “Auto 1” otteen stabiilisuusvirheiden histogrammi



Kuva 17: Kappaleen “Muki 1” otteen stabiilisuusvirheiden histogrammi



Kuva 18: Kappaleen “Pullo 1” otteen stabiilisuusvirheiden histogrammi



Kuva 19: Kappaleen “Vasara 1” otteen stabiilisuusvirheiden histogrammi

6 YHTEENVETO

Työssä koestettiin monikerrosperseptronin, radiaalikantafunktioverkon, tukivektoriregression ja relevanssivektoriregression soveltuvuutta robottisormien nivelkulmien ja otteen stabiilisuuden mallintamisessa. Kokeita varten menetelmille ohjelmoitiin testiympäristö Matlabiin. Mallinnuksessa käytettiin 12 erilaista kolmiulotteista kappaletta neljästä eri kappalekategoriasta. Kokeista kirjattiin tarkastelua varten sormien nivelkulmien virheet, stabiilisuuden virheet ja laskenta-ajat. Lopuksi saatuja tuloksia verrattiin keskenään.

Työn tuloksien perusteella tukivektoriregressio osoittautui tarkaksi ja nopeaksi mallinnusmenetelmäksi. Tukivektoriregressio oli myös ainoa menetelmä, joka kykeni rajallisella muistimäärällä käyttämään kappaleen datan täysimääräisenä. Muut työssä käytetyt menetelmät eivät yltäneet SVR:n tasolle. Monikerrosperseptroni oli epätarkin menetelmä. Tukivektoriregression hyvien tulosten johdosta sen koetuloksia analysoitiin muita menetelmiä pidemmälle. Tukivektoriregression laskemista sormien nivelkulma- ja stabiilisuusarvoista piirrettiin virhehistogrammit. Sormien nivelkulma-arvojen virhehistogrammi vaikutti normaalijakautuneelta. Stabiilisuusarvojen virhehistogrammeista nähtiin SVR:n laskevan stabiilisuusarvon oikeata isommaksi yleisimmässä virhetapauksessa.

Aikataulu poikkesi suunnitellusta, sillä työn vaiheet olivat yhdestä kahteen viikkoa jäljessä. Menetelmien kuvaamiseen ja tulosten laskemiseen kului odotettua enemmän aikaa. Loppuun jätetty viimeistelyaika käytettiin kokonaan, joskaan työssä ei koettu suuria odottamattomia ongelmia. Tulevaisuudessa olisi mielenkiintoista tutkia tukivektoriregression virheen ja laskenta-ajan reagointia erilaisiin datamääriin, mitä tässä työssä ei tehty ajan loppumisen vuoksi.

LÄHTEET

[Bic00] Bicchi, A., Hands for dexterous manipulation and robust grasping: a difficult road toward simplicity. *Institute of Electrical and Electronics Engineers Transactions on Robotics and Automation*, 16, sivut 652–662.

[Bis06] Bishop, C. M., *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[Fle10] Fletcher, T., Relevance vector machines explained, 2010. [Verkkodokumentti, viitattu 7.10.2011]. Saatavilla: http://www.tristanfletcher.co.uk/RVM%20Explained.pdf.

[GCDA09] Goldfeder, C., Ciocarlie, M., Dang, H. ja Allen, P. K., The columbia grasp database. *Proceedings of the 2009 Institute of Electrical and Electronics Engineers international conference on Robotics and Automation*, ICRA'09, Piscataway, NJ, USA, 2009, Institute of Electrical and Electronics Engineers Press, sivut 3343–3349.

[Hay99] Haykin, S., *Neural Networks, A Comprehensive Foundation*. Pearson Education, toinen painos, 1999.

[HCL03] Hsu, C.-W., Chang, C.-C. ja Lin, C.-J., A practical guide to support vector classification, 2003. [Verkkodokumentti, viitattu 15.9.2011]. Saatavilla: http://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf.

[HRK08] Huebner, K., Ruthotto, S. ja Kragic, D., Minimum volume bounding box decomposition for shape approximation in robot grasping. *Robotics and Automation, 2008. ICRA 2008. Institute of Electrical and Electronics Engineers International Conference on*, sivut 1628–1633.

[KS96] Kröse, B. ja Smagt, P., *An introduction to neural networks*. University of Amsterdam, 8. painos, 1996.

[Lam97] Lampinen, J., Advances in neural network modeling. *Proc. of TOOLMET'97, Tool Environments and Development Methods for Intelligent Systems*,

Yliniemi, L. ja Juuso, E., toimittajat, osa 1, Helsinki University of Technology, 1997, sivut 28–36. Plenary presentation.

[Mur10] Murphy, K., Pmtk3: Probabilistic modeling toolkit for matlab/octave, version 3, 2010. Saatavilla: http://code.google.com/p/pmtk3/.

[Ros58] Rosenblatt, F., The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 6, sivut 386–408.

[SK08] Siciliano, B. ja Khatib, O., toimittajat. *Springer Handbook of Robotics*, sivut 671–672. Springer, Berlin, Heidelberg, 2008.

[SL05] Shibin, Q. ja Lane, T., Multiple kernel learning for support vector regression, 2005. [Verkkodokumentti, viitattu 7.10.2011]. Saatavilla: http://www.cs.unm.edu/ treport/tr/05-12/QiuLane.pdf.

[TF03] Tipping, M. ja Faul, A., Fast marginal likelihood maximisation for sparse bayesian models. *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, Tammikuu 2003, sivut 3–6.

[Tip00] Tipping, M., The relevance vector machine. *Advances in Neural Information Processing Systems*, 12, sivut 652–658.

[Tip01] Tipping, M., Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1, sivu 211–244.

[Vap99] Vapnik, V., An overview of statistical learning theory. *Institute of Electrical and Electronics Engineers Transaction on Neural Networks*, 5, sivut 988–999.

LIITE 1. Kokeiden Matlab - koodi

```
%%    Niko Reunanen
% ====================
% niko.reunanen@lut.fi

%% Alkuasetukset
% Kappaleista muodostetaan taulukko helppoa iterointia varten
kappaleet = num2cell(ones(4,3,1)*0);
for i = 1:3,
    kappaleet{1,i} = autot4{1,i};
    kappaleet{2,i} = mukit10{1,i};
    kappaleet{3,i} = pullot8{1,i};
    kappaleet{4,i} = vasarat4{1,i};
end

% Kumpi puolikas kappaledatasta, 1 vai 2?
palanen = 2;
% Valitse menetelma (1=MLP, 2=RBFN, 3=SVR, 4=RVR)
menetelma = 4;
% Asetetaan menetelmien parametrit. Kokeissa kaytetyt parametrit etsittiin
% tyon aikaisemmassa vaiheessa.
%% [Multilayer Perceptron]:
%   Ensimmaisen piilokerroksen neuronimaara
    n1 = 95;
%   Toisen piilokerroksen neuronimaara
    n2 = round(n1 / 5);
%% [Radial Basis Function Network]:
%   Kantafunktion leveysparametrin arvo
    leveys_rbfn = 0.7;
%% [Support Vector Regression]:
%   epsilon, haettu virhetaso
    virheraja = 0.02525;
%   Kompromissiarvo
    c_arvo = 3;
%   Kantafunktion leveysparametrin arvo
```

```
35 g_arvo = 5;
36 %% [Relevance Vector Regression]:
37 %   Kantafunktion leveysparametrin arvo
38     leveys = 3;
39
40 %% Alkuasetukset
41 % Ristiinvalidoinnin kertaluku, parametrihaussa 2 ja kokeissa 10
42 ristiinvalidoinnin_kertaluku = 10;
43 % Menetelmien kappalekohtaiset nivelkulmavirheet
44 kappaletulokset_sormet = num2cell(ones(4,3,1)*0);
45 % Menetelmien kappalekohtaiset stabiilisuusvirheet
46 kappaletulokset_stabiilisuus = num2cell(ones(4,3,1)*0);
47 % Menetelmien kappalekohtaiset koulutus- ja laskenta-ajat
48 koulutusajat = num2cell(ones(4,3,1)*0);
49 % Koulutetut RVR - mallit yhdelle kappaleelle koulutuksen aikana
50 rvr_malli = cell(4,1);
51 % Koulutetut SVR - mallit yhdelle kappaleelle koulutuksen aikana
52 svr_malli = cell(4,1);
53 % Seurataan kaytettyjen relevanssivektorien keskimaaraa: nivelkulmat ja
54 % stabiilisuus
55 rv_maara = num2cell(ones(4,3,4)*0);
56 % Seurataan kaytettyjen tukivektorien keskimaaraa: nivelkulmat ja
57 % stabiilisuus
58 sv_maara = num2cell(ones(4,3,4)*0);
59 % Valiaikaisen sisallon pitaja SVR - ja RVR - malleille
60 laskennat = cell(4,1);
61 % Parametrit LibSVM - koodikirjaston SVM - koulutukseen
62 svr_asetukset = ['-s 3 -h 0 -t 2 -g ', num2str(g_arvo),' -c ',...
63     num2str(c_arvo), ' -e ', num2str(virheraja)];
64
65 % Laskentoja varten otetaan kayttoon edistymisikkuna, joka nayttaa
66 % graafisesti kuinka paljon tyota on jaljella.
67 edistyminen = 1;
68 edistymisikkuna=waitbar(0,'Lasketaan');
69
70 %% Menetelmien kaytto
71 % Iteroidaan kategoriat ja kappaleet
```

```
72 for kategoria_id = 1:4,
73     for kappale_id = 1:3,
74
75         % Valitaan ristiinvalidointidata ja otetaan myohempaa kayttoa
76         % varten stabiilisuuden maksimiarvo talteen.
77         [ristidata, maxi_stabarvo] = kappaleenRistivalidointidata(...
78             kappaleenPalanen(kappaleet{kategoria_id, kappale_id},...
79             palanen), ristiinvalidoinnin_kertaluku);
80
81         laskenta_aika = 0;
82         koulutus_aika = 0;
83         % a = ulostulo
84         for a = 1:ristiinvalidoinnin_kertaluku,
85             data = cell(2,2);
86             % Paivitetaan nykyinen tyotilanne
87             waitbar( edistyminen / 120, edistymisikkuna,...
88                 sprintf('KAT: %g  KAP: %g  a: %g',...
89                 kategoria_id, kappale_id, a) );
90             edistyminen = edistyminen + 1;
91
92             % b = sisaantulo
93             for b = 1:ristiinvalidoinnin_kertaluku,
94                 % Rakennetaan ristiinvalidointidata jossa yksi osa dataa
95                 % testaa ja loput kouluttavat.
96                 if (a == b),
97                     data{2,1} = ristidata{a,1};
98                     data{2,2} = ristidata{a,2};
99                 else
100                    data{1,1} = vertcat(data{1,1}, ristidata{b,1});
101                    data{1,2} = vertcat(data{1,2}, ristidata{b,2});
102                end
103            end
104
105            %% Relevance Vector Regression
106            if (menetelma == 4),
107                % Jokaiselle ulostulolle koulutetaan oma RVR
108                for i = 1:4,
```

```
109                     tic
110                     rvr_malli{i} = rvmFit(data{1,1}, data{1,2}(:,i),...
111                         'kernelFn',...
112                         @(X1, X2)kernelRbfGamma(X1, X2, leveys));
113                     koulutus_aika = koulutus_aika + toc;
114 
115                     tic
116                     laskennat{i} = rvmPredict(rvr_malli{i}, data{2,1});
117                     laskenta_aika = laskenta_aika + toc;
118 
119                     rv_maara{kategoria_id,kappale_id,i} =...
120                     rv_maara{kategoria_id,kappale_id,i} +...
121                     size(rvr_malli{i}.Relevant,1);
122                 end
123 
124                 % Matriisi nivelkulmien ennustetuista arvoista
125                 rvr_palautus = [laskennat{1} laskennat{2} laskennat{3}];
126 
127                 % Lasketaan MSE ennustetuista arvoista oikeisiin arvoihin
128                 % nahden
129                 virhe = mse( (rvr_palautus - data{2,2}(:,1:3) ) *2.5133);
130                 kappaletulokset_sormet{kategoria_id,kappale_id} =...
131                    kappaletulokset_sormet{kategoria_id,kappale_id} + virhe;
132 
133                 % Vektori stabiilisuuksien ennustetuista arvoista
134                 rvr_palautus = laskennat{4};
135 
136                 % Alle 0.1 arvoiset stabiilisuudet muutetaan nollaksi
137                 rvr_palautus(rvr_palautus < 0.1) = 0;
138 
139                 % Lasketaan MSE stabiilisuuksien ennustetuista arvoista
140                 % oikeisiin nahden
141                 virhe = mse((rvr_palautus-data{2,2}(:,4))*maxi_stabarvo);
142                 kappaletulokset_stabiilisuus{kategoria_id,kappale_id} =...
143                    kappaletulokset_stabiilisuus{kategoria_id,kappale_id}...
144                    + virhe;
145             end
```

```
146
147             %% Support Vector Regression
148             if (menetelma == 3),
149                 % Jokaiselle ulostulolle koulutetaan oma SVR
150                 for i = 1:4,
151                     tic
152                     svr_malli{i} = svmtrain(data{1,2}(:,i),data{1,1},...
153                         svr_asetukset);
154                     koulutus_aika = koulutus_aika + toc;
155
156                     sv_maara{kategoria_id,kappale_id,i} =...
157                     sv_maara{kategoria_id,kappale_id,i} +...
158                     svr_malli{i}.totalSV;
159
160                     tic
161                     [t1,¬,¬] = svmpredict(data{2,2}(:,i), data{2,1},...
162                         svr_malli{i});
163                     laskenta_aika = laskenta_aika + toc;
164
165                     laskennat{i} = t1;
166                 end
167                 svr_palautus = [laskennat{1} laskennat{2} laskennat{3}];
168                 virhe = mse( (svr_palautus - data{2,2}(:,1:3)) * 2.5133);
169                 kappaletulokset_sormet{kategoria_id,kappale_id} =...
170                    kappaletulokset_sormet{kategoria_id,kappale_id} + virhe;
171
172                 svr_palautus = laskennat{4};
173
174                 % Alle 0.1 arvoiset stabiilisuudet muutetaan nollaksi
175                 svr_palautus(svr_palautus < 0.1) = 0;
176
177                 virhe = mse((svr_palautus-data{2,2}(:,4))*maxi_stabarvo);
178                 kappaletulokset_stabiilisuus{kategoria_id,kappale_id} =...
179                    kappaletulokset_stabiilisuus{kategoria_id,kappale_id}...
180                    + virhe;
181
182             end
```

```
183
184             %% Radial Basis Function Network
185             if (menetelma == 2),
186                 tic;
187                 rbfn = newrb( data{1,1}', data{1,2}', 0.001, leveys_rbfn,...
188                     200, 100000);
189                 koulutus_aika = koulutus_aika + toc;
190
191                 tic;
192                 tmp_1 = rbfn(data{2,1}');
193                 virhe = mse( (tmp_1(1:3,:)' - data{2,2}(:,1:3)) * 2.5133 );
194                 kappaletulokset_sormet{kategoria_id,kappale_id} =...
195                    kappaletulokset_sormet{kategoria_id,kappale_id} + virhe;
196
197                 % Alle 0.1 arvoiset stabiilisuudet muutetaan nollaksi
198                 tmp_2 = tmp_1(4,:)';
199                 tmp_2(tmp_2 < 0.1) = 0;
200                 tmp_1(4,:) = tmp_2';
201
202                 virhe = mse( (tmp_1(4,:)' - data{2,2}(:,4))*maxi_stabarvo);
203                 kappaletulokset_stabiilisuus{kategoria_id,kappale_id} =...
204                    kappaletulokset_stabiilisuus{kategoria_id,kappale_id}...
205                    + virhe;
206
207                 laskenta_aika = laskenta_aika + toc;
208             end
209
210             %% Multilayer Perceptron
211             if (menetelma == 1),
212                 mlp = feedforwardnet([n1 n2]);
213                 mlp.trainParam.goal = 0.0001;
214                 mlp.trainParam.showWindow = 0;
215                 mlp.trainParam.epochs = 1000;
216                 mlp.performParam.regularization = 0.75;
217
218                 tic;
219                 mlp_koulutettu = trainlm(mlp, data{1,1}', data{1,2}');
```

```
220                 koulutus_aika = koulutus_aika + toc;
221
222                 tic;
223                 tmp_1 = sim(mlp_koulutettu, data{2,1}');
224                 virhe = mse( (tmp_1(1:3,:)' - data{2,2}(:,1:3)) * 2.5133 );
225                 kappaletulokset_sormet{kategoria_id,kappale_id} =...
226                    kappaletulokset_sormet{kategoria_id,kappale_id} + virhe;
227
228
229                 % Alle 0.1 arvoiset stabiilisuudet muutetaan nollaksi
230                 tmp_2 = tmp_1(4,:)';
231                 tmp_2(tmp_2 < 0.1) = 0;
232                 tmp_1(4,:) = tmp_2';
233
234                 virhe = mse( (tmp_1(4,:)'-data{2,2}(:,4))*maxi_stabarvo );
235                 kappaletulokset_stabiilisuus{kategoria_id,kappale_id} =...
236                    kappaletulokset_stabiilisuus{kategoria_id,kappale_id}...
237                    + virhe;
238                 laskenta_aika = laskenta_aika + toc;
239             end
240         end
241
242         if (menetelma == 4),
243             for i = 1:4,
244                 rv_maara{kategoria_id,kappale_id,i} =...
245                 rv_maara{kategoria_id,kappale_id,i} /...
246                 ristiinvalidoinnin_kertaluku;
247             end
248         end
249
250         if (menetelma == 3),
251             for i = 1:4,
252                 sv_maara{kategoria_id,kappale_id,i} =...
253                 sv_maara{kategoria_id,kappale_id,i} /...
254                 ristiinvalidoinnin_kertaluku;
255             end
256         end
```

```
257
258         % Nivelkulmien ristiinvalidoinnin keskimaarainen normalisoitu virhe
259         kappaletulokset_sormet{kategoria_id,kappale_id} =...
260             kappaletulokset_sormet{kategoria_id,kappale_id}...
261             / ristiinvalidoinnin_kertaluku;
262
263         % Stabiilisuuden ristiinvalidoinnin keskimaarainen normalisoitu
264         % virhe
265         kappaletulokset_stabiilisuus{kategoria_id,kappale_id} =...
266             kappaletulokset_stabiilisuus{kategoria_id,kappale_id}...
267             / ristiinvalidoinnin_kertaluku;
268
269         % Menetelman keskimaarainen koulutusaika ristiinvalidoinnissa
270         koulutus_aika = koulutus_aika / ristiinvalidoinnin_kertaluku;
271         % Menetelman keskimaarainen laskenta-aika ristiinvalidoinnissa
272         laskenta_aika = laskenta_aika / ristiinvalidoinnin_kertaluku;
273
274         koulutusajat{kategoria_id,kappale_id} =...
275             [koulutus_aika laskenta_aika];
276     end
277 end
278 %% Tuloksien palautus
279 % Suljetaan ja poistetaan edistymisikkuna
280 delete(edistymisikkuna);
281 % Menetelman kappalekohtaisten nivelkulmien normalisoimaton virhe
282 virhe_sum = sum(sum(cell2mat(kappaletulokset_sormet)));
283 % Menetelman kappalekohtaisten stabiilisuuksien normalisoimaton virhe
284 virhe_sum_stabiilisuus = sum(sum(cell2mat(kappaletulokset_stabiilisuus)));
```