Harjaregression ja lasson bayesiläinen tulkinta

Emilia Kalliokoski

Pro gradu -tutkielma
Kesäkuu 2024

Tarkastajat:
FT Janne Kujala
Prof. Kari Auranen

MATEMATIIKAN JA TILASTOTIETEEN LAITOS

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

Emilia Kalliokoski: Harjaregression ja lasson bayesiläinen tulkinta
Pro gradu -tutkielma, 35 s., 16 liites.
Tilastotiede, Tilastotieteen linja
Kesäkuu 2024

Tilastollisessa mallintamisessa tavoitteena on sovittaa aineistoon malli, joka kuvaa mahdollisimman hyvin aineiston selittävien muuttujien suhdetta aineiston vastemuuttujaan. Hyvä malli noudattaa opetuksessa käytettyä aineistoa niin hyvin, että sen tekemät päätelmät uusista havainnosta ovat mahdollisimman tarkkoja, mutta ei kuitenkaan niin hyvin, että malli oppii aineistossa esiintyvää satunnaisuutta. Näin saattaa käydä esimerkiksi silloin, kun sovitettava malli on hyvin monimutkainen. Mallin sanotaan olevan ylisovittunut, kun verrattain yksinkertaisempi malli tuottaisi parempia, vähemmän virheellisiä tuloksia. Mallin ylisovittumista voidaan välttää erilaisia regularisointimenetelmillä, jotka pyrkivät yksinkertaistamaan sovitettavaa mallia.

Lineaarinen regressiomalli olettaa nimensä mukaisesti, että aineiston selittävien muuttujien suhde vastemuuttujaan on lineaarinen. Lineaarista mallia sovitettaessa pyritään muodostamaan estimaatit regressiokertoimille, jotka kuvaavat selittävien muuttujien suhdetta vastemuuttujaan. Yleinen tapa estimoida regressiokertoimet on pienimmän neliösumman menetelmä, joka perustuu jäännösneliösumman minimointiin.

Harjaregressio ja lasso ovat menetelmiä lineaarisen regressiomallin regularisoimiseksi. Ne minimoivat sakotettua jäännösneliösummaa, jossa jäännösneliösumman lisäksi minimoidaan sakkotermiä, joka kutistaa mallin regressiokertoimia. Regularisoinnin voimakkuutta säätelee sakkotermin säätöparametri. Kun kertoimien vaikutusta rajoitetaan, malli yksinkertaistuu ja ylisovittuminen vähenee. Harjaregressio säilyttää kaikki selittävät muuttujat mallissa, mutta lasso voi kutistaa osan kertoimista tasan nollaksi.

Harjaregressiota ja lassoa vastaavat regressiokertoimet voidaan muodostaa myös bayesiläisittäin MAP-estimoinnilla. Harjaregressiota vastaavat MAP-estimaatit saadaan, kun mallin priorijakaumana käytetään normaalijakaumaa, jonka keskiarvo on nolla ja varianssi riippuu regularisoinnin säätöparametrista. Lassoa vastaavat MAP-estimaatit saadaan, kun mallin priorijakaumana käytetään Laplace-jakaumaa, jonka lokaatio on nolla ja skaala riippuu säätöparametrista.

Asiasanat: lineaarinen regressio, ylisovitus, regularisointi, harjaregressio, lasso, bayesiläinen MAP-estimointi.

Sisällys

1 Johdanto

Tilastollisen oppimisen tyypillinen ydinongelma on löytää tapa hyödyntää jo olemassa olevaa, havaittua dataa vielä tuntemattoman datan ymmärtämiseen [1][2]. Mielenkiintona voi olla melkeinpä mitä tahansa osakkeen hinnan muutosten ennustamisesta siihen, että aivosähkökäyrädatan perusteella luokitellaan, kärsiikö tutkittava potilas epilepsiasta. Jotta dataa voidaan hyödyntää uusien havaintojen ennustamiseen ja niistä tehtävään päättelyyn, datasta pitää löytää suhteita, rakenteita ja keskinäisiä vaikutuksia. Jos yrityksellä on suuri määrä työntekijöitä, onko yrityksen osakkeen hinta silloin korkeampi? Millä aivoaaltojen taajuuksilla havaitut poikkeamat viittaavat eniten epileptisiin kohtauksiin? Jotta dataa voidaan hyödyntää, sille muodostetaan tilastollinen malli, jolla kuvataan dataan vaikuttavien muuttujien suhteita toisiinsa.

Karkeasti sanottuna tilastollisen mallin tulee olla tarkka ja yleistettävä. Tarkka malli tekee oikeellisia päätelmiä: malli kuvaa sen opettamisessa käytettyä dataa niin hyvin, että se osuu tuntemattomia havaintoja estimoitaessa mahdollisimman lähelle kyseisen havainnon todellista arvoa. Mallia ei ole kuitenkaan hyvä opettaa noudattamaan sen opetuksessa käytettyä dataa *liian* hyvin. Kaikessa datassa on aina mukana myös satunnaisuutta, joka ei suoraan johdu mistään aineistoa varsinaisesti selittävästä muuttujasta. Jos malli sovitetaan opetuksessa käytettävään dataan niin hyvin, että se oppii myös aineiston satunnaisvaihtelua, sen tulokset eivät ole enää yleistettäviä. Toisin sanoen, kun tietyn datan satunnaisvaihtelua oppinutta mallia sovelletaan ennustamaan uutta dataa, jossa ei yksinkertaisesti ole samaa satunnaisvaihtelua, se ei suoriudu tehtävästään hyvin. Voisikin sanoa, että tilastollista mallia ei ole tarkoitus opettaa tulkitsemaan nimenomaan jotain tiettyä aineistoa mahdollisimman hyvin, vaan oppimisen kohteena tulisi olla ilmiö, joka on synnyttänyt kyseisen aineiston. [1][2]

Yksi yleinen ja suoraviivainen tilastollisen mallin estimointimenetelmä on lineaarinen regressiomallintaminen. Yleinen lineaarinen malli olettaa nimensä mukaisesti, että aineiston selittävien muuttujien suhde mielenkiinnon kohteena olevaan vasteeseen on lineaarinen. Jokaisen selittävän muuttujan vaikutusta ja sen voimakkuutta säätelee regressiokerroin, ja regressiokertoimet ovatkin lineaarisessa mallintamisessa estimoitavat tuntemattomat parametrit. [1][2]

Sovittamalla lineaarista mallia aineistoon ottamatta kantaa siihen, ovatko kaikki aineiston selittävät muuttujat yhteydessä tutkittavaan vasteeseen, tai onko muuttujien vaikutus merkittävää, päädytään usein tilanteeseen, jossa mallin tuottama päättely ei ole kovinkaan yleistettävää. Lineaarista mallia voidaan pyrkiä saattamaan muotoon, joka tuottaa yleistettävämpiä tuloksia erilaisin menetelmin, joista yksi kategoria on regularisointimenetelmät. Regularisointimenetelmät pyrkivät kutistamaan lineaarisen mallin regressiokertoimia, jolloin niihin liittyvien selittävien muuttujien vaikutus pienenee. Kun muuttujien vaikutus mallissa pienenee, malli kokonaisuudessaan yksinkertaistuu ja se tuottaa yleistettävämpiä tuloksia. [1][2]

Tilastollisia malleja muodostaessa olennaisena kysymyksenä on sekä mallin yleisen muodon valinta että tietynlaisen mallin parametrien estimointi. Tilastollisella päättelyllä on kaksi suurta koulukuntaa, frekventistinen ja bayesiläinen päättely, jotka eroavat toisistaan erityiseesti siinä, miten ne tulkitsevat parametreja. Frekven-

tistisen tilastotieteen klassinen ajattelutapa on, että estimoitavat parametrit ovat tuntemattomia mutta vakioita. Parametrilla on siis jokin tietty ”oikea” arvo, jolle pyritään muodostamaan mahdollisimman hyvä estimoitu arvo. Koska parametri on vakio, sille ei voida suoraan olettaa mitään todennäköisyysjakaumaa. Sille voidaan kuitenkin muodostaa otosjakauma sen perusteella, millaisia arvoja tutkittavaa parametria vastaava tunnusluku saa eri otoksissa, jotka kerätään tutkittavasta populaatiosta. [3][4]

Bayesiläisessä tilastotieteessä tutkittavaa tuntematonta parametria ei pidetä vakiona vaan satunnaismuuttujana. Parametrilla ei siis ajatella olevan mitään varsinaista todellista arvoa, ja siihen liittyy aina jotain epävarmuutta. Koska parametria pidetään satunnaismuuttujana, sille voidaan kuitenkin määritellä todennäköisyysjakauma, jonka perusteella tulkitaan parametrin käyttäytymistä ja sen mahdollisia arvoja. Tämä jakauma määritellään jo ennen aineiston keruuta, eli se kuvaa oikeastaan tutkittavasta parametrista tiedettävää ennakkotietoa tai tehtyjä ennakkooletuksia. Lopulliset päätelmät tehdään yhteisjakaumasta, joka koostuu sekä tästä ennakkotiedosta että kerätyn aineiston antamasta informaatiosta. [3][4]

Tässä työssä käsitellään lineaarisen regressiomallin regularisointia, ja regularisointimenetelmistä käsitellään erityisesti harjaregressiota ja lassoa. Lineaarista mallintamista ja sen regularisointia tulkitaan ensin frekventistisestä näkökulmasta, mutta työn erityistarkoituksena on antaa käsiteltäville menetelmille motivaatio ja tulkinta myös bayesiläisittäin. Tilastollisen oppimisen peruskäsitteitä ja lineaarista regressiota käsitellään luvussa 2 ja regularisointimenetelmiä luvussa 3. Käsiteltäviä menetelmiä tulkitaan bayesiläisittäin luvussa 4, ja luvussa 5 menetelmiä sovelletaan esimerkkiaineistoon.

2 Tilastollinen oppiminen

Tässä luvussa pohjustetaan tutkielman aihetta tilastollisen oppimisen perusteilla. Luvun lähteenä käytetään kahta tilastollisen oppimisen oppikirjaa: *Elements of Statistical Learning* [1] ja *Introduction to Statistical Learning* [2].

Tilastollinen oppiminen on keskeistä monilla tilastotieteen aloilla, ja sillä pyritään tekemään päätelmiä tai ennusteita jonkin aineiston pohjalta. Oppimiseen käytetään mallia f, joka kuvaa aineiston selittävien muuttujien $\mathbf{X} = (X_1, X_2, \ldots, X_p)$ suhdetta vastemuuttujaan Y. Vastemuuttujaa voidaan kutsua myös esimerkiksi selitettäväksi tai ennustettavaksi muuttujaksi. Suhde voidaan kirjoittaa yleisessä muodossa

$$Y = f(\mathbf{X}) + \varepsilon,$$

jossa ε kuvaa satunnaisia virhetermejä. Virhetermit ovat riippumattomia muuttujista $\mathbf{X}$ ja niiden odotusarvon oletetaan olevan nolla. Malli f on usein tuntematon, mutta sitä voidaan estimoida eri tilastollisen oppimisen menetelmillä.

2.1 Ennustaminen ja päättely

Kun mallia f estimoidaan, mielenkiintona voi olla joko ennustaminen tai päättely, tai niiden yhdistelmä. Yleisesti mallin estimoinnissa tarkoituksena on löytää aineiston perusteella mallille f estimaatti $\hat{f}$, joka soveltuu aineistoon mahdollisimman hyvin: toisin sanoen, tavoitteena on estimoida malli $\hat{f}$, jolle pätee $Y \approx \hat{f}(\mathbf{X})$ millä tahansa aineiston havainnolla. Ennustamisen ja päättelyn eri tavoitteiden takia mallia saatetaan estimoida ja lopullinen malli voidaan valita eroavin perustein.

Ennustettaessa selittävät muuttujat $\mathbf{X}$ ovat usein helposti saatavilla, mutta vastemuuttujat Y eivät ole. Koska ennuste perustuu odotusarvoon, se ei riipu virhetermeistä ε, jolloin virhetermit supistuvat pois. Ennusteet $\hat{Y}$ saadaan siis yhtälöstä

$$\hat{Y} = \hat{f}(\mathbf{X}).$$

Ennustettaessa tavoitteena ei välttämättä ole estimoida mallille mahdollisimman tarkkaa ja todellisuutta vastaavaa muotoa: riittää, että estimoitu malli muodostaa mahdollisimman tarkkoja ennusteita.

Päättelyssä mielenkiintona on selittävien muuttujien $\mathbf{X}$ ja vastemuuttujan Y välinen suhde, eikä niinkään tulevien havaintojen ennustaminen. Tarkemmin mielenkiintona voi olla ymmärtää, miten selittävien muuttujien vaihtelu vaikuttaa vasteen arvoon, tai mitkä selittävät muuttujat ylipäätään vaikuttavat vasteeseen, ja kuinka suuri vaikutus jokaisella yksittäisellä muuttujalla on. Tällöin mallin $\hat{f}$ muoto on tärkeää tietää täsmällisesti.

2.2 Parametriset ja epäparametriset menetelmät

Tilastollisen oppimisen menetelmät voidaan karkeasti jakaa parametrisiin ja epäparametrisiin menetelmiin sen perusteella, mitä menetelmän tuottaman malli muodosta oletetaan. Parametrisissa menetelmissä lähdetään liikkeelle oletuksesta, että malli f on jotain tiettyä muotoa. Kun oletus mallin muodosta on tehty, riittää, että mallin parametrit estimoidaan käyttäen jotain siihen soveltuvaa menetelmää

siten, että parametrit kuvaavat aineistoa mahdollisimman tarkasti. Tällöin mallin estimointiongelma yksinkertaistuu: on yleisesti helpompaa estimoida jonkin tietyn mallin parametreja, kuin täysin mielivaltaista kokonaista mallia.

Yksinkertainen esimerkki parametrisesta menetelmästä on olettaa, että estimoitava malli on lineaarinen eli muotoa

$$f(\mathbf{X}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \ldots \beta_p X_p.$$

Tällöin mallin parametrit $\beta_0, \beta_1, \ldots, \beta_p$ voidaan estimoida aineiston perusteella käyttämällä esimerkiksi pienimmän neliösumman menetelmää.

Parametrisilla menetelmillä ei yleensä pystytä muodostamaan mallia $\hat{f}$, joka vastaisi täysin mallin f todellista muotoa. Jos valittu malli on kaukana todellisesta mallista, saadut estimaatit eivät ole hyviä, eikä niiden perusteella voida tehdä luotettavia päätelmiä. Tätä ongelmaa voidaan pyrkiä välttämään käyttämällä joustavampia malleja, jotka toisaalta voivat sopia kuvaamaan monia mahdollisia mallin f muotoja, mutta toisaalta vaativat myös usein suurempaa määrää parametreja, joita täytyy estimoida. Liian monimutkaiset mallit voivat johtaa ongelmaan, jossa malli tosiasiassa noudattaa liikaa aineistossa esiintyvää vaihtelua. Ongelmaa kutsutaan ylisovittumiseksi, ja siihen palataan tarkemmin myöhemmin tässä tutkielmassa.

Epäparametrisissa menetelmissä ei tehdä tarkkoja oletuksia mallin f muodosta, vaan menetelmät pyrkivät löytämään estimaatin $\hat{f}$, joka noudattaa mahdollisimman tarkasti havaintoja, mutta ei kuitenkaan ole liian karkea tai epätasainen. Koska oletusta mallin f todellisesta muodosta ei tehdä, epäparametrisin menetelmin sovitetut mallit voivat sopia kuvaamaan monia erilaisia mahdollisia mallin f muotoja. Koska epäparametriset menetelmät eivät yksinkertaista mallin estimointia pelkkien mallin parametrien estimointiin, tarkan estimaatin $\hat{f}$ muodostamiseksi tarvitaan toisaalta paljon suurempi määrä havaintoja kuin parametrisia menetelmiä käytettäessä.

2.3 Ohjattu ja ohjaamaton oppiminen

Tilastollisen oppimisen ongelmat voidaan jakaa karkeasti kahteen luokkaan: ohjattuun ja ohjaamattomaan oppimiseen. Ohjatussa oppimisessa jokaisella havaitulla selittävien muuttujien arvolla on vastaava vastemuuttujan arvo. Tavoitteena on sovittaa malli, joka yhdistää vastemuuttujan selittäviin muuttujiin, minkä kautta voidaan joko ennustaa tulevia havaintoja tai ymmärtää paremmin selittävien ja selitettävien muuttujien välistä suhdetta.

Ohjaamattomassa oppimisessa havaituilla selittävien muuttujien arvoilla ei ole vastaavia vastemuuttujia. Tavoitteena on löytää aineistosta rakenne, joka selittää eri havaintojen suhdetta toisiinsa tai esimerkiksi tapa esittää aineisto vain muutaman selittävän muuttujan tai ulottuvuuden avulla.

2.4 Luokittelu ja regressio

Tilastollisen oppimisen ongelmat voidaan jakaa kahteen luokkaan myös muuttujien tyyppien perusteella: luokitteluun ja regressioon. Kvalitatiivisten muuttujien arvot ovat jonkin kategorian tai luokan eri arvoja, esimerkiksi sukupuoli tai veriryhmä.

Kvantitatiiviset muuttujat saavat numeerisia arvoja, esimerkiksi henkilön ikä tai osakkeen hinta.

Tilastollisen oppimisen ongelmia, joissa vastemuuttuja on kvalitatiivinen, kutsutaan luokitteluongelmiksi. Jos vastemuuttuja on kvantitatiivinen, kyseessä on regressio-ongelma. Selittävien muuttujien tyypillä ei yleensä ole merkitystä tilastollisen oppimisen menetelmiä käytettäessä: analyysien suorittaminen voi kuitenkin vaatia muuttujien, yleensä kvalitatiivisten, esitystavan muuttamista. Kvalitatiiviset muuttujat voidaan muuttaa dummy- eli indikaattorimuuttujiksi, jolloin muuttujien eri tasot pisteytetään arvoilla 0 ja 1 siten, että niitä voidaan käyttää kvantitatiivisten muuttujien tapaan analyyseissa.

2.5 Mallin arviointi

2.5.1 Opetus-, validointi- ja testijoukko

Kun halutaan arvioida tilastollisen mallin ennustekykyä, on tapana jakaa aineisto satunnaisesti kolmeen joukkoon: opetus-, validointi- ja testijoukkoon. Aineisto jaetaan yleensä siten, että opetusjoukkoon kuuluu enemmän havaintoja kuin validointi- ja testijoukkoihin. Jako voisi esimerkiksi olla käyttää 50 % havainnoista opetusjoukkona, 25 % validointijoukkona ja 25 % testijoukkona.

Opetusjoukkoa käytetään nimensä mukaan mallin opettamiseen: sen avulla sovitetaan mallin parametrit. Sovitettuja malleja arvioidaan ja mallien parametreja voidaan säädellä validointijoukon avulla. Sovitettuja malleja käytetään ennustamaan validointijoukon havaintoja, ja vertailemalla eri malleja ja niiden tarkkuutta pyritään valitsemaan malli, joka antaa tarkimmat tulokset. Validointijoukon avulla valittua lopullista mallia arvioidaan vielä täysin erillisellä testijoukolla, jotta saadaan harhaton arvio valitun mallin tarkkuudesta.

2.5.2 Virheen arviointi

Jotta tilastollisen oppimisen menetelmiä ja niiden tuottamia malleja voidaan arvioida, tarvitaan tapa mitata, miten hyvin niiden tuottamat ennusteet vastaavat havaittua aineistoa. Tarvitaan siis keino laskea, kuinka paljon estimoitu vaste eroaa sitä vastaavan havainnon todellisesta vasteesta.

Regressio-ongelmissa yleisin virheen mitta on keskineliövirhe (*mean squared error, MSE*)

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}\left(\mathbf{x}_i\right)\right)^2, \tag{1}$$

jossa y_i on vasteen havaittu arvo ja $\hat{f}\left(\mathbf{x}_i\right)$ estimoidun mallin antama estimoitu vaste, jotka vastaavat i:nnettä havaintoa. Keskineliövirhe kasvaa neliöllisesti: kun vasteen estimaatti ja todellinen arvo ovat lähellä toisiaan, virhe on pieni, mutta eron suuretessa myös virhe suurenee nopeasti.

Keskineliövirhe (1) lasketaan käyttämällä opetusaineistoa, jota käytetään mallin sovittamiseen aineistoon: tarkemmin sitä voisi siis kutsua opetusvirheeksi. Yleensä kiinnostuksen kohteena on kuitenkin ennemmin niiden ennusteiden tarkkuus, jotka saadaan, kun malli sovitetaan aikaisemmin tuntemattomaan testiaineistoon. Toisin

sanoen, tavoitteena on löytää malli, jolle pätee mahdollisimman hyvin $\hat{f}(\mathbf{x}_0) \approx y_0$, jossa $(\mathbf{x}_0, y_0)$ on havainto testiaineistosta, jota ei ole käytetty mallinnusmenetelmän opettamiseen. Tarkoituksena olisi siis valita malli, joka tuottaa mahdollisimman pienen testivirheen. Testivirhe saadaan laskemalla keskimääräinen neliövirhe testihavainnoista $(\mathbf{x}_0, y_0)$:

$$Ave\left(y_0 - \hat{f}(\mathbf{x}_0)\right)^2.$$

Luokitteluongelmissa virheen laskemiseen tarvitaan eri tapa, sillä vastemuuttujien arvot eivät ole numeerisia. Tyypillinen virheen mitta on virheosuus (*error rate*), joka kuvaa estimaatin $\hat{f}$ tekemien virheiden osuutta kaikista luokitteluista:

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i),$$

jossa $\hat{y}_i$ on ennustettu luokan arvo i:nnelle havainnolle, kun ennustamiseen käytetään mallia $\hat{f}$, ja $I(y_i \neq \hat{y}_i)$ on indikaattorimuuttuja, joka saa arvon 1, jos $y_i \neq \hat{y}_i$ ja arvon 0, jos $y_i = \hat{y}_i$. Eli, jos $I(y_i \neq \hat{y}_i) = 0$, i:nnes havainto luokiteltiin oikein, ja muutoin havainto luokiteltiin väärin. Tämä virheosuus lasketaan käyttämällä opetusaineistoa, joten sitäkin voi kutsua regressiotilannetta vastaten opetusvirheosuudeksi. Vastaavasti kiinnostavampaa on laskea testivirheosuus käyttämällä testihavaintoja (x_0, y_0):

$$Ave\left(I(y_0 \neq \hat{y}_0)\right).$$

Tässä $\hat{y}_0$ on ennustettu luokan arvo, joka vastaa selittävän muuttujan arvoa $\mathbf{x}_0$.

2.5.3 Harha ja varianssi

Testivirheeseen vaikuttaa olennaisesti kaksi ominaisuutta: harha ja varianssi. Varianssi kuvaa tässä yhteydessä, kuinka paljon $\hat{f}$ muuttuisi, jos sen estimointiin käytettäisiin eri opetusjoukkoa. Yleensä joustavammilla ja monimutkaisemmilla tilastollisilla menetelmillä on suurempi varianssi, koska pienetkin muutokset aineistossa johtavat merkittävään muutokseen mallissa $\hat{f}$. Suuri varianssi voi siis johtaa siihen, että malli oppii liikaa aineiston omaa satunnaista vaihtelua, joka ei välttämättä ole aineiston taustalla olevan ilmiön kannalta oleellista.

Harha mittaa virhettä, joka syntyy, kun monimutkaista ongelmaa mallinnetaan paljon yksinkertaisemmalla mallilla. Esimerkiksi lineaarisissa malleissa tehdään hyvin vahva oletus yksinkertaisesta lineaarisesta suhteesta selittävien ja selitettävän muuttujan välillä, vaikka käytännössä kovinkaan moni todellinen ongelma ei pelkisty niin yksinkertaiseksi. Usein monimutkaisemmat menetelmät johtavatkin pienempään harhaan.

Voidaan osoittaa, että testivirheen odotusarvolle pätee hajotelma

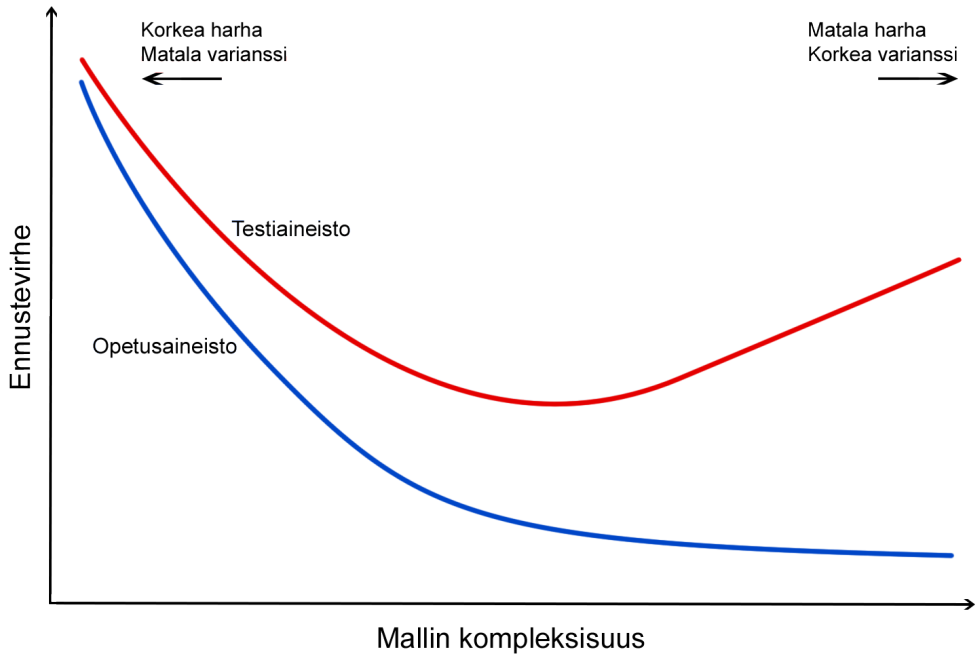
$$E\left(y_0 - \hat{f}(\mathbf{x}_0)\right)^2 = \mathrm{Var}\left(\hat{f}(\mathbf{x}_0)\right) + \left(\mathrm{Bias}\left(\hat{f}(\mathbf{x}_0)\right)\right)^2 + \mathrm{Var}(\varepsilon).$$

Testivirheen odotusarvo tarkoittaa keskimääräistä testivirhettä, joka saataisiin estimoitaessa toistuvasti mallia f käyttäen suurta määrää eri opetusjoukkoja. Hajotelmasta nähdään, että testivirheen odotusarvon minimoimiseksi vaaditaan sekä pieni varianssi että pieni harha.

Varianssi ja harha eivät kuitenkaan kulje käsi kädessä: kun mallin kompleksisuus kasvaa, sen varianssi kasvaa ja harha pienenee. Varianssin ja harhan välinen suhteellinen muutosnopeus määrittelee, kasvaako vai pieneneekö testivirhe. Yleensä mallin harha pienenee aluksi nopeammin kuin varianssi kasvaa, mutta jossain kohtaa mallin joustavuuden lisääntyessä sillä ei ole enää suurta merkitystä harhaan, mutta varianssi alkaa lisääntyä merkittävästi. Tällöin myös testivirheen arvo alkaa kasvaa. Ilmiötä voidaan kutsua harha-varianssi-kompromissiksi. On yleensä melko yksinkertaista löytää malli, jolla on joko hyvin suuri varianssi ja pieni harha, tai pieni varianssi ja suuri harha. Haasteena onkin löytää menetelmä ja malli, joilla molemmat ominaisuudet jäävät pieniksi.

Virhe opetusaineistolla käyttäytyy eri tavalla. Opetusvirheellä on tapana laskea, kun mallin kompleksisuus kasvaa. Testivirhe taas yleensä laskee ensin ja tietyn pisteen jälkeen alkaa taas kasvamaan. Vaikka siis opetusvirhe olisikin pieni monimutkaisella mallilla, sen tuottamat tulokset eivät yleisty opetusjoukon ulkopuolelle kovinkaan hyvin. Opetus- ja testiaineistojen virheiden käyttäytymistä visualisoidaan kuvassa 1.

Tilanteissa, joissa verrattain yksinkertaisempi malli tuottaisi pienemmän testivirheen, käytetään termiä *ylisovittuminen*. Kun malli sovitetaan opetusaineistoon lähes täydellisesti, päädytään tyypillisesti tilanteeseen, jossa malli sovittuu opetusaineistossa ilmentyvään satunnaiseen vaihteluun, eikä niinkään koko aineiston taustalla olevaan todelliseen ilmiöön. Kun mallia, joka on opetettu havaitsemaan satunnaisvaihtelun aiheuttamaa kohinaa, sovelletaan opetusaineiston ulkopuolelle, se ei suoriudu tehtävästään kovinkaan hyvin, koska tätä kohinaa ei yksinkertaisesti ole olemassa testiaineistossa.



Kuva 1: Opetusaineiston ja testiaineiston virheet mallin kompleksisuuden funktiona. Kun malli on tarpeeksi monimutkainen, sen harha ei enää merkittävästi pienene, mutta varianssi lisääntyy: tällöin kokonaisvirhe testiaineistolla alkaa taas kasvaa.

2.6 Lineaarinen regressio

2.6.1 Lineaarinen malli

Yleinen lineaarinen malli olettaa, että selittävien muuttujien suhde vastemuuttujaan on lineaarinen, tai vähintään, että lineaarinen malli on järkevä approksimaatio kyseiselle suhteelle. Lineaarinen regressiomalli on muotoa

$$\begin{aligned} Y = f(\mathbf{X}) + \varepsilon &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots \beta_p X_p + \varepsilon \\ &= \beta_0 + \sum_{j=1}^{p} X_j \beta_j + \varepsilon, \end{aligned}$$

jossa β_0 on vakiotermi, joka kuvaa vastemuuttujan Y arvoa, kun $X = 0$, ja $\beta_1, \ldots, \beta_p$ ovat jokaista selittävää muuttujaa X_j vastaavat regressiokertoimet, jotka kuvaavat kunkin selittävän muuttujan vaikutusta vastemuuttujaan. Voidaan tulkita, että β_j on keskimääräinen vaikutus vastemuuttujaan Y, kun sitä vastaavan selittävän muuttujan X_j arvo kasvaa yhdellä, ja muut selittävät muuttujat pysyvät vakioina. Regressiokertoimet β_j ovat tuntemattomia, ja ne ovatkin estimoinnissa mielenkiinnon kohteena. Virhetermi ε kuvaa havaitun vastemuuttujan arvon ja mallin tuottaman ennusteen erotusta.

Lineaarinen malli voidaan kirjoittaa myös matriisimuodossa

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_{11} & \ldots & x_{1p} \\ \vdots & & \vdots \\ x_{n1} & \ldots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

eli

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}. \tag{2}$$

Vastemuuttujan ja selittävien muuttujien lineaarisen suhteen lisäksi lineaarisen mallin käyttö vaatii muitakin oletuksia. Selittävät muuttujat eivät korreloi voimakkaasti keskenään, ja niiden havaitut arvot ovat kiinteitä ja vakioita. Virhetermit ε_i noudattavat normaalijakaumaa, jonka odotusarvo on 0 ja varianssi on vakio — toisin sanoen virhetermit ovat homoskedastisia. Virhetermit eivät myöskään korreloi keskenään.

2.6.2 PNS-menetelmä

Parametrejä $\beta_0, \ldots, \beta_p$ estimoitaessa käytetään havaintoja, jotka ovat muotoa $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$. Jokainen vektori $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^T$ kuvaa selittävien muuttujien arvoja i:nnen havainnon kohdalla. Yleisin estimointimenetelmä on pienimmän neliösumman (PNS) menetelmä, jossa kertoimet $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)^T$ valitaan siten, että ne minimoivat jäännösneliösumman (*residual sum of squares, RSS*):

$$\mathrm{RSS} = \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{n} x_{ij} \beta_j \right)^2. \tag{3}$$

Jäännösneliösumma on siis yksi tapa mitata vastemuuttujan arvon y_i ja mallin estimoiman arvon $f(\mathbf{x}_i)$ välistä eroa, joka pyritään pitämään mahdollisimman pienenä.

Hyödyntäen yleisen lineaarisen mallin matriisimuotoista esitystä (2) jäännösneliösumma voidaan kirjoittaa toisella tavalla:

$$\mathrm{RSS} = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}),$$

jossa $\mathbf{y} = (y_1, \ldots, y_n)^T$ on vektori vastemuuttujan arvoista. Kun yhtälö derivoidaan parametrin $\boldsymbol{\beta}$ suhteen ja derivaatan nollakohta ratkaistaan, yhtälön minimikohdaksi ja suurimman uskottavuuden estimaatiksi $\hat{\boldsymbol{\beta}}$ saadaan

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}.$$

Kun saatu suurimman uskottavuuden estimaatti sijoitetaan takaisin yleisen lineaarisen mallin yhtälöön, estimoiduiksi vastemuuttujan arvoiksi saadaan

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}.$$

2.6.3 Lineaarisen mallin arviointi

Lineaarisen regressiomallin sopivuutta arvioidaan tyypillisesti käyttämällä jäännösvirhettä (*residual standard error, RSE*) ja selitysastetta R^2.

Jäännösvirhe estimoi lineaarisen mallin virhetermien ε keskihajontaa. Sitä voidaan ajatella mallin yhteensopivuuden puutteen (*lack of fit*) mittarina: jos mallin tuottamat ennusteet ovat hyvin lähellä vastemuuttujien todellisia arvoja, jäännösvirhe on pieni ja voidaan todeta, että malli sopii aineistoon hyvin. Jäännösvirhe lasketaan kaavalla

$$\mathrm{RSE} = \sqrt{\frac{1}{n-p-1}\mathrm{RSS}} = \sqrt{\frac{1}{n-p-1}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2},$$

jossa RSS on mallin jäännösneliösumma (3).

Koska jäännösvirhe riippuu vastemuuttujasta, sen tulkinta ei ole aina yksiselitteistä. Selitysaste R^2 kuvaa mallin selittämän varianssin osuutta — kuinka suuren prosenttiosuuden vastemuuttujan vaihtelusta malli pystyy selittämään. R^2 saakin aina arvoja nollan ja yhden välillä, eikä se ole riippuvainen vastemuuttujan skaalasta. Selitysaste R^2 saadaan kaavalla

$$R^2 = \frac{\mathrm{TSS} - \mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2},$$

jossa RSS on mallin jäännösneliösumma (3), ja TSS on mallin kokonaisneliösumma.

Selitysaste mittaa mallin yhteensopivuutta (*goodness of fit*). Kokonaisneliösumma TSS kuvaa kaikkea vaihtelua yhteensä, jota vastevektorissa $\mathbf{y}$ esiintyy. Jäännösneliösumma RSS kuvaa vaihtelua, jota lineaarinen regressiomalli ei ole pystynyt selittämään. Kun näiden kahden luvun välinen suhde on pieni, ja selitysaste R^2 saa siten suuren arvon, voidaan todeta mallin sopivan aineistoon hyvin.

Selitysasteen R^2 tulkintaa vaikeuttaa se, että jäännösneliösumma RSS pienenee aina, kun malliin lisätään uusia selittäviä muuttujia, vaikka kyseiset muuttujat itsessään olisivat vain heikosti yhteydessä vastemuuttujaan. Tästä syystä mallin muuttujien lisääntyessä myös R^2 suurenee aina, vaikka suureneminen olisikin marginaalista.

Korjattu R^2 ottaa selitysasteen laskennassa huomioon mallin käyttämän selittävien muuttujien lukumäärän. Se lasketaan kaavalla

$$\text{Korjattu } R^2 = 1 - \frac{\text{RSS}/\left(n-p-1\right)}{\text{TSS}/\left(n-1\right)},$$

jossa p on mallin selittävien muuttujien määrä. Ottamalla muuttujien määrä huomioon selitysastetta laskiessa vältytään ainakin teoriassa ns. ylimääräisten muuttujien lisäämiseltä malliin. Kun malliin lisätään muuttujia, jotka eivät juurikaan pienennä jäännösneliösummaa, muuttujien määrän p kasvaessa myös $\text{RSS}/\left(n-p-1\right)$ suurenee, jolloin korjattu R^2 vuorostaan pienenee.

3 Regularisointi

Lineaarisen mallin sovittaminen käyttäen pienimmän neliösumman menetelmää antaa usein hyvän lähtökohdan vastemuuttujan ja selittävien muuttujien välisen suhteen selvittämiselle. Lineaarisen mallintamisen tuloksia voidaan kuitenkin tyypillisesti parantaa käyttämällä vaihtoehtoisia sovitusmenetelmiä, jotka parantavat ennusteiden tarkkuutta ja mallin tulkittavuutta.

PNS-menetelmän tuottamilla estimaateilla on tyypillisesti pieni harha, mutta varianssi voi olla hyvinkin suuri. Tämä pätee etenkin tilanteissa, joissa havaintojen määrä n ei ole kovinkaan paljon suurempi kuin mallin muuttujien määrä p. Jos muuttujia on enemmän kuin havaintoja, PNS-menetelmää ei voida edes käyttää, sillä se ei anna tässä tilanteessa yksiselitteisiä tuloksia. Ylipäätään kaikkien saatavilla olevien selittävien muuttujien sisällyttäminen regressiomalliin lisää malliin turhaa kompleksisuutta, koska joukossa on luultavasti muuttujia, joilla ei ole suhdetta vastemuuttujaan, tai joiden vaikutus vastemuuttujaan on hyvin pieni.

Rajoittamalla estimoitujen regressiokertoimien vaikutusta mallissa, tai jopa jättämällä muuttujia kokonaan pois mallista, mallin varianssi voi pienentyä hyvinkin selkeästi. Vaikka mallin muuttujien vaikutuksen tai niiden kokonaismäärän vähentäminen tarkoittaa toki myös mallin harhan lisääntymistä, lopputuloksena on silti tyypillisesti ennustevirheen pieneneminen, koska mallin varianssi pienenee harhan kasvamista voimakkaammin.

Puhtaasti pienimmän neliösumman menetelmän tuottamaa mallia käyttämällä törmätäänkin usein ylisovittumisen ongelmaan: sovittamalla aineistoon täysi lineaarinen malli saadaan testiaineistolla huonompia tuloksia kuin käyttämällä yksinkertaisempaa mallia. On olemassa erilaisia menetelmiä, joiden avulla voi säädellä lineaariseen malliin sisällytettävien muuttujien määrää ja vaikutusta. Tämän työn mielenkiintona on menetelmät, jotka pyrkivät kutistamaan estimoituja regressiokertoimia mallin varianssin pienentämiseksi. Tätä kutsutaan *regularisoinniksi*.

Tämän luku perustuu samoihin tilastollisen oppimisen oppikirjoihin kuin luku 2: *Elements of Statistical Learning* [1] ja *Introduction to Statistical Learning* [2].

3.1 Kutistavat menetelmät

Kutistavat menetelmät säilyttävät lineaarisessa mallissa kaikki p selittävää muuttujaa, mutta pyrkivät regularisoimaan niille PNS-menetelmää käyttämällä estimoituja kertoimia kutistamalla niitä kohti nollaa. Tämä tapahtuu siten, että tavanomaisen jäännösneliösumman minimoimisen sijaan regressiokertoimet estimoidaan minimoimalla *sakotettua* jäännösneliösummaa

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{n} x_{ij}\beta_j\right)^2 + \lambda S\left(\boldsymbol{\beta}\right) = \mathrm{RSS} + \lambda S\left(\boldsymbol{\beta}\right), \tag{4}$$

jossa $\lambda S\left(\boldsymbol{\beta}\right)$ on sakkotermi, joka muodostuu sakkofunktiosta $S\left(\boldsymbol{\beta}\right)$ ja säätöparametrista $\lambda \geq 0$. PNS-menetelmän tavoin kutistavat menetelmät pyrkivät estimoimaan kertoimet $\beta_1, \ldots, \beta_p$ siten, että ne sopivat malliin mahdollisimman hyvin minimoimalla jäännösneliösummaa. Tämän lisäksi sakkotermin avulla kertoimet pyritään

pitämään mahdollisimman lähellä nollaa käyttämällä sakkofunktiota, joka saa pienen arvon, kun regressiokertoimet ovat pieniä. Sakkofunktiona käytetään tyypillisesti normifunktiota, joka jollain tapaa summaa regressiokertoimia yhteen — käytännössä sitä voi ajatella esimerkiksi regressiokertoimista muodostuvan vektorin pituutena. Säätöparametri λ määrittelee, kuinka voimakasta kutistaminen on: mitä suurempi parametrin arvo on, sitä voimakkaampaa kutistaminen on. Eri λ:n arvot tuottavat eri joukon estimaatteja, minkä takia sopivan säätöparametrin valinta onkin tärkeää.

Parametrin λ arvon valintaan voi käyttää esimerkiksi K-kertaista ristiinvalidointia. Aineisto jaetaan pienempiin samankokoisiin osiin, joita on K kappaletta: jokaista osaa käytetään kerran validointiaineistona, ja muita $K-1$ osaa opetusaineistona. Muulla aineistolla opetettu malli sovitetaan K:nnenteen osaan, josta lasketaan keskineliövirhe. Menettely toistetaan K kertaa, joista koostetaan keskimääräinen keskineliövirhe. Tämä lasketaan käyttäen useita eri mahdollisia säätöparametrin λ arvoja, joiden keskuudesta valitaan parametrin arvo, jonka tuottama virhe on pienin.

Kutistavia menetelmiä käyttäessä muuttujien havainnot on tapana ensin normalisoida odotusarvolle nolla ja varianssille yksi, koska menetelmien ratkaisut eivät ole skaalaltaan ekvivariantteja. Toisin sanoen kutistavien menetelmien tuottamat estimaatit regressiokertoimille voivat muuttua hyvinkin voimakkaasti riippuen selittävien muuttujien skaalasta. Kun muuttujat normalisoidaan, niiden skaala on sama. Kun muuttujat normalisoidaan, vakiotermi β_0, jonka estimaatti on $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$, voidaan jättää sovituksen ulkopuolelle, sillä sen arvoksi tulee nolla.

Sakotetun jäännösneliösumman ongelma (4) voidaan muotoilla myös tavalla, josta käy intuitiivisemmin ilmi sakkotermin luoma rajoite regressiokertoimien koolle:

$$\arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{n} x_{ij}\beta_j \right)^2 \text{ ehdolla } S(\boldsymbol{\beta}) \leq s, \tag{5}$$

jossa parametri s on jokin vakio, joka tuottaa samat ratkaisut kuin säätöparametri λ yhtälössä (4). Parametri s toimii rajana sille, kuinka suuren arvon sakkofunktio $S(\boldsymbol{\beta})$, jota normifunktiona voi käytännössä ajatella vektorin $(\beta_1, \ldots, \beta_p)^T$ pituutena, voi saada. Toisin sanoen kutistavat menetelmät pyrkivät PNS-menetelmän tavoin löytämään regressiokertoimet, jotka tuottavat mahdollisimman pienen jäännösneliösumman, mutta niiden yhteispituus ei saa ylittää asetettua rajaa s.

3.1.1 Harjaregressio

Harjaregressio (*ridge regression*) on kutistava menetelmä, joka käyttää sakotetun jäännösneliösumman minimoinnissa sakkofunktiona vektorin etäisyyttä origosta mittaavaa l_2-normia

$$||\boldsymbol{\beta}||_2 = \sqrt{\sum_{j=1}^{p} \beta_j^2} = \sqrt{\beta_1^2 + \ldots + \beta_p^2}.$$

Kun l_2-normin neliöjuuren alla oleva osa sijoitetaan sakotettuun jäännösneliösummaan (4), harjaregression ratkaisut saadaan minimoimalla

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{n} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2. \tag{6}$$

Menetelmässä sakkofunktiona toimii siis käytännössä regressiokertoimien neliöiden summa. Kun $\lambda = 0$, harjaregression estimaatit vastaavat PNS-menetelmän estimaatteja, ja kun $\lambda \to \infty$, kertoimet kutistuvat sekä kohti nollaa että myös toisia kertoimia. Yksikään kerroin ei kuitenkaan kutistu tismalleen nollaksi (paitsi jos $\lambda = \infty$, jolloin kaikki kertoimet ovat nollia). Harjaregressio säilyttää siis aina mallin kaikki selittävät muuttujat.

3.1.2 Lasso

Lassoregressio (*LASSO; least absolute shrinkage and selection operator*) on harjaregressiota muistuttava kutistava menetelmä, jossa käytetään l_2-normin sijasta l_1-normin

$$||\boldsymbol{\beta}||_1 = \sqrt{\sum_{j=1}^{p} |\beta_j|} = \sqrt{|\beta_1| + \ldots + |\beta_p|}$$

mukaista sakkofunktiota jäännösneliösumman minimoinnissa (4):

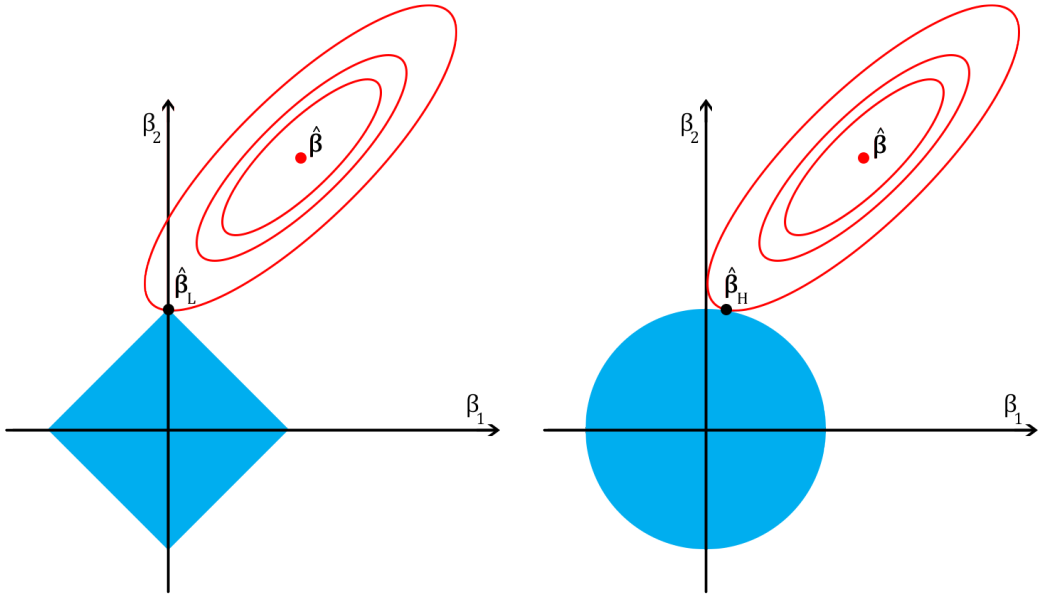
$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{n} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|. \tag{7}$$

Lassossa sakkofunktio on siis regressiokertoimien itseisarvojen summa.

Kuten harjaregressiossa, säätoparametrin λ suuretessa lassoregressiokertoimet pienenevät kohti nollaa — erona kuitenkin on, että kun λ on riittävän suuri, osa kertoimista kutistuu lassoregresiossa tismalleen nollaksi. Käytännössä lassoregressio siis tekee regularisoinnin lisäksi muuttujavalintaa: se valitsee kaikkien mahdollisten muuttujien joukosta pienemmän osajoukon muuttujia, jotka pystyvät edelleen kuvaamaan selittävien muuttujien ja vastemuuttujan välistä suhdetta riittävän hyvin.

Harjaregression ja lasson geometrinen tulkinta (kuva 2) auttaa ymmärtämään, miksi lasso voi antaa ratkaisun, jossa osa estimaateista on tismalleen nollia. Sakotettu jäännösneliösumma voidaan muotoilla vaihtoehtoisesti tavalla, jossa jäännösneliösumma minimoidaan tietyllä ehdolla (5); harjaregressiossa ehto on $\beta_1^2 + \ldots + \beta_j^2 \leq s$ ja lassossa $|\beta_1| + \ldots + |\beta_p| \leq s$ jollain parametrin arvolla s. Kun $p = 2$, harjaregression antama mahdollisten ratkaisujen joukko muodostaa ympyrän akselien β_1 ja β_2 leikkauskohdan ympärille, ja lasso vastaavasti timantin muotoisen ratkaisujoukon. Jäännösneliösumma muodostaa pienimmän neliösumman estimaatin $\hat{\boldsymbol{\beta}}$ ympärille ellipsin muotoisia tasa-arvokäyriä, joista jokaisen ellipsin pisteille jäännösneliösumma on yhtä suuri — mitä kauempana tasa-arvokäyrä on PNS-estimaatista, sitä suurempi jäännösneliösumma on.

Yhtälöstä (5) voidaan tulkita, että harjaregression ja lasson ratkaisut saadaan ensimmäisestä pisteestä, jossa jäännösneliösumman tasa-arvokäyrä leikkaa mahdollisen ratkaisujen joukon reunan. Koska lasson mahdollisella ratkaisujoukolla on terävät kulmat akseleilla, leikkauskohta on usein juuri akselien kohdalla, joissa yksi tai useampi β_j saa arvoksi nollan. Mitä enemmän mallissa on muuttujia, sitä enemmän ratkaisujoukossa on teräviä kulmia eli mahdollisuuksia, että osa kertoimista saa arvoksi nollan. Koska harjaregression ratkaisujoukko on aina pyöreä, leikkauspiste ei samalla tavalla osu tyypillisesti juuri akseleille.



Kuva 2: Kun $p = 2$, lasso antaa timantin muotoisen (sinisellä vasemmalla) ja harjaregressio pyöreän ratkaisujoukon (sinisellä oikealla). Ratkaisujoukon ja jäännösneliösumman tasa-arvokäyrien (punaisella) ensimmäinen leikkauspiste antaa minimointiongelman (5) ratkaisun: $\hat{\boldsymbol{\beta}}_L$ merkitsee lasson ratkaisua ja $\hat{\boldsymbol{\beta}}_H$ harjaregression ratkaisua. Kuvan piirtämisessä on käytetty mallina lähteiden [1][2] vastaavia kuvia.

3.2 Menetelmien vertailua

PNS-estimaateilla on usein pieni harha mutta suuri varianssi, jolloin pienikin muutos opetusaineistossa voi johtaa suuriin eroihin estimoiduissa regressiokertoimissa. Harja- ja lassoregressio ovat näissä tilanteissa tehokas apukeino: kun säätöparametri λ saa tarpeeksi suuria arvoja, mallin joustavuus laskee, jolloin harha toki nousee hieman, mutta varianssi laskee sitäkin enemmän. Tällöin sekä harhasta että varianssista riippuva testivirhe kaiken kaikkiaan pienenee, ja harja- ja lassoregressiomallin ennustavuus on PNS-mallia parempi.

Pienimmän neliösumman menetelmää ei voi myöskään käyttää, kun muuttujien määrä p on suurempi kuin havaintojen määrä n, koska sillä ei saada tällöin yksiselitteistä ratkaisua. Harjaregressiota ja lassoa voi kuitenkin soveltaa myös tilanteissa, joissa $p > n$.

Vaikka sekä harjaregressio että lasso yleisesti tuottavat parempia tuloksia kuin PNS-menetelmä, ne soveltuvat parhaiten hieman erilaisiin tilanteisiin. Harjaregressiomalleilla saavutetaan tavanomaisesti hieman parempi ennustetarkkuus kuin lassolla — toisaalta lassomalli, jossa osa muuttujista jää kokonaan pois, on yksinkertaisempi ja siten helpommin tulkittava kuin harjaregressiomalli.

Lasso soveltuukin parhaiten tilanteisiin, joissa vain pieni osa kaikista muuttujista on todellisesti merkittäviä havaintojen taustalla olevan ilmiön selittämiseen, koska tällöin ns. ylimääräisten muuttujien pudottaminen pois mallista ei haittaa sen ennustamiskykyä. Koska harjaregressio kutistaa regressiokertoimia myös kohti toisiaan sen lisäksi, että ne kutistuvat kohti nollaa, se soveltuu paremmin tilanteisiin,

joissa suuri osa kaikista muuttujista on merkittäviä, ja todelliset regressiokertoimet olisivat kooltaan samaa luokkaa.

On kuitenkin lähes mahdotonta tietää, kuinka suuri osuus aineiston selittävistä muuttujista on yhteydessä vastemuuttujaan ennen mallin sovittamista. Tästä syystä eri mallinnusmenetelmiä voi testata aineistoon ennen lopullista mallin sovitusta käyttämällä esimerkiksi ristiinvalidointia.

4 Bayesiläinen tulkinta

Työssä esiteltyjä tilastollisia menetelmiä on tähän mennessä lähestytty klassisen, frekventistisen tilastotieteen näkökulmasta. Tutkittavaa populaatiota kuvaava parametri on tuntematon, mutta vakio — toisin sanoen parametrilla ajatellaan olevan jokin tietty todellinen arvo, mutta arvoa ei tiedetä. Koska parametri on jokin vakio, sillä ei voida ajatella olevan todennäköisyysjakaumaa, joka kertoisi, millä todennäköisyydellä parametri saisi mitäkin arvoja. Parametrin arvoa pyritäänkin siis estimoimaan keräämällä populaatiosta otos, jolle lasketaan tunnusluku, joka vastaa kiinnostuksen kohteena olevaa koko populaatiota koskevaa parametria. Sen jälkeen, kun populaatiosta on kerätty (hypoteettisesti) kaikki mahdolliset otokset ja niille on laskettu otostunnusluvut, nämä tunnusluvut muodostavat tutkittavalle parametrille otosjakauman. Otosjakauman perusteella voidaan siten tehdä päätelmiä tutkittavan parametrin todennäköisimmästä arvosta jollain tietyllä luottamustasolla. [3][4]

Bayesiläisessä päättelyssä epävarmuutta tutkittavan parametrin todellisesta arvosta pyritään kuvaamaan jakaumien avulla. Jos tuntematonta parametria pidetäänkin vakion sijaan satunnaismuuttujana, sille voidaan määritellä jokin todennäköisyysjakauma jo ennen kuin populaatiosta on kerätty yhtäkään havaintoa. Tämä jakauma kuvaa ennakkotietoa parametrin käyttäytymisestä ja sen mahdollisista arvoista. Kun tähän ennakkotietoon lopulta yhdistetään kerätyn otoksen tuoma informaatio, niiden yhteisjakaumasta voidaan tehdä päätelmiä tutkittavan parametrin arvosta sillä ehdolla, että on kerätty juuri havaittu aineisto. Tutkittavan parametrin käyttäytymisestä tehdään siis ennakko-oletuksia, joita ikään kuin arvioidaan uudelleen sen perusteella, mitä kerätty otos kertoo parametrin mahdollisista arvoista. Lopulliset päätelmät parametrista tehdään sekä kerätyn aineiston tuoman havaitun informaation että ennakko-oletusten tuoman havaitsemattoman informaation perusteella. [3][4]

Tässä luvussa käsitellään lineaarista regressiomallinnusta ja erityisesti sen regularisointia bayesiläisestä näkökulmasta.

4.1 Bayesiläinen päättely

Bayesiläinen tilastotiede perustuu Bayesin teoreemaan ehdollisista todennäköisyyksistä. Kun merkitään tapahtumien A ja B todennäköisyyksiä vastaavasti *P(A)* ja *P(B)*, tapahtuman B todennäköisyys ehdolla A saadaan kaavalla [3][5]

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)} = \frac{P(A \mid B)\, P(B)}{P(A)}.$$

Merkitään tutkittavaa tuntematonta parametria θ ja havaintoa y, jotka vastaavasti noudattavat jakaumia $p(\theta)$ ja $p(y)$. Parametria θ kuvaavaa ennakkotietoa $p(\theta)$ kutsutaan *priorijakaumaksi*. Otokseen perustuva mallin parametrien *uskottavuusfunktio* $p(y \mid \theta)$ kuvaa havainnon y otantajakaumaa ehdolla parametrit, näiden parametrien funktiona tarkasteltuna. Parametrin θ todennäköisyyttä havainnolla y ehdollistettuna kuvaava jakauma saadaan Bayesin teoreemaa soveltamalla:

$$p(\theta \mid y) = \frac{p(\theta, y)}{p(y)} = \frac{p(y \mid \theta)\, p(\theta)}{p(y)}, \tag{8}$$

jossa $p(\theta, y)$ on θ:n ja y:n yhteisjakauma. Parametrista θ saatua tietoa havainnolla y ehdollistettuna $p(\theta \mid y)$ kutsutaan *posteriorijakaumaksi.* [4]

Jakauma $p(y)$ on havaitun otoksen *marginaalijakauma.* Jos θ on diskreetti, $p(y) = \sum_\theta p(\theta)\, p(y \mid \theta)$. Jos θ on jatkuva, $p(y) = \int p(\theta)\, p(y \mid \theta)\, d\theta$. Koska otos y on kiinteä, eikä sen marginaalijakauma riipu parametrista θ, sitä voidaan pitää vakiona [4]. Posteriorijakaumasta (8) käytetäänkin usein sen verrannollista muotoa, josta marginaalijakauma on jätetty pois [3]:

$$p(\theta \mid y) \propto p(y \mid \theta)\, p(\theta)\,. \tag{9}$$

4.1.1 Priorijakauman valinta

Priorijakauman valinta on luonteeltaan hyvin subjektiivista. Koska priori valitaan ennen yhdenkään havainnon tekemistä tutkittavasta populaatiosta ja valinta perustuu pelkästään ilmiötä tutkivan henkilön ennakkokäsityksiin siitä, mitä mahdollisia arvoja tuntematon parametri θ voi saada, yhden henkilön valitsema priori ei välttämättä ole sama kuin toisen henkilön valitsema priori [3]. Priorijakauman valinta onkin kriittisin — ja kritisoidun — osa bayesiläistä tilastollista analyysiä [5][6]. Saatavilla oleva ennakkotieto tutkittavasta ilmiöstä on käytännössä harvoin niin kattavaa, että yhden tietyn, parhaiten sopivan priorijakauman valinta usean, ainakin jossain määrin sopivalta vaikuttavan jakauman joukosta olisi mahdollista.

Yksi tekniikka priorijakauman valintaan on käyttää konjugaattiprioria. Jos F on luokka uskottavuuksia $p(y \mid \theta)$, ja P on luokka parametrin θ priorijakaumia $p(\theta)$, niin luokka P on luokan F konjugaattiperhe, jos $p(\theta \mid y) \in P$ kaikille $p(y \mid \theta) \in F$ ja $p(\theta) \in P$ [4]. Jos luokaksi P valitaan kaikki mahdolliset jakaumat, se on aina luokan F konjugaattiperhe. Konjugaattipriorin hyötyihin päästään käsiksi, kun P on luonnollinen konjugaattiperhe. Tällöin priorin ja uskottavuuden riippuvuus parametreista on samanlaista muotoa.

Yksinkertaistettuna konjugaattipriorin käyttäminen tarkoittaa, että priorijakaumaperhe valitaan uskottavuuden eli havaintojen jakauman perusteella. Uskottavuudesta riippuen mahdollisia konjugaattiprioreita voi olla useita. Esimerkiksi binomijakauman konjugaattipriori on betajakauma, mutta normaalijakaumalla on useita eri konjugaattiprioreita. Normaalijakauman konjugaattipriorin voi valita sen perusteella, mitkä jakauman parametreista ovat tunnettuja. [4]

Konjugaattipriorin käyttämisellä on laskennallisia etuja, sillä sen seurauksena posteriorijakaumalla on parametrinen muoto. Konjugaattipriorin käyttö soveltuu myös tilanteisiin, joissa ennakkotietoa on saatavilla vain rajallisesti tai tieto ei ole kovinkaan täsmällistä. Useita yksinkertaisia konjugaattiprioreita voidaan myös käyttää muodostamaan monimutkaisempia malleja. Luonnollisia konjugaattiprioreita on kuitenkin tyypillisesti vain jakaumilla, jotka kuuluvat eksponentiaaliperheeseen. [4][5]

Jos ennakkotietoa on saatavilla vain hyvin vähän tai ei ollenkaan, priorijakauman subjektiivinen määrittely voi olla hyvin hankalaa tai jopa mahdotonta. Tällaisissa tilanteissa priorin vaikutus posteriorijakaumaan halutaan tyypillisesti minimoida, ja prioriksi voidaan tällöin valita epäinformatiivinen priori, joka johdetaan ainoasta saatavilla olevasta informaatiosta eli otosjakaumasta $p(y \mid \theta)$.

Yksi tapa johtaa epäinformatiivinen priori on käyttää Jeffreysin prioria [4][5]. Menetelmä perustuu Fisherin informaatioon, joka määritellään yksiulotteisissa ta-

pauksissa seuraavasti:

$$J(\theta) = E\left[\left(\frac{d \log p(y \mid \theta)}{d\theta}\right)^2\right] = -E\left[\frac{d^2 \log p(y \mid \theta)}{d\theta^2}\right].$$

Jeffreysin prioriksi saadaan täten

$$p(\theta) \propto [J(\theta)]^{1/2}.$$

Priorijakauman perustamisen Fisherin informaatioon on perusteltua, koska Fisherin informaatio on pätevä indikaattori sille, kuinka paljon informaatiota havainnot antavat parametrista θ. Jeffreysin priorissa ne parametrin θ arvot, joille $J(\theta)$ on suurempi, ovat todennäköisempiä. [5]

Jeffreysin priori yleistyy myös moniulotteisiin tapauksiin: tällöin Fisherin informaatio voidaan esittää matriisimuodossa [5]

$$[J(\theta)]_{i,j} = -E\left[\frac{d^2}{d\theta_i d\theta_j} \log p(y \mid \theta)\right],$$

kun $\theta \in \mathbb{R}^N$ ja $i, j = 1, \ldots, N$. Jeffreysin priori on tällöin

$$p(\theta) \propto [\det(J(\theta))]^{1/2}.$$

Kun parametri θ on moniulotteinen, Jeffreysin priorin käyttö voi kuitenkin johtaa epäjohdonmukaisuuksiin ja ristiriitaisiin tuloksiin. Se soveltuukin parhaiten tapauksiin, joissa parametri θ on yksiulotteinen. [4][5]

4.2 MAP-estimointi

Bayesiläisessä päättelyssä ydinajatuksena on tehdä päätelmiä tuntemattomasta parametrista θ käyttämällä posteriorijakaumaa $p(\theta \mid y)$, joka sisältää kaiken parametrista θ saatavilla olevan tiedon yhdistämällä ennakkotiedon parametrista $p(\theta)$ havaitun aineiston tuomaan tietoon parametrista $p(y \mid \theta)$ [5][6]. Yksi tapa hyödyntää posteriorijakaumaa on etsiä piste-estimaatti, joka maksimoi posteriorijakauman. Tätä estimaattia kutsutaan *maximum a posteriori* -estimaatiksi eli MAP-estimaatiksi, joka vastaa posteriorijakauman moodia [5][7]:

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} p(\theta \mid y) = \arg\max_{\theta} p(y \mid \theta)\, p(\theta).$$

MAP-estimointi vastaa periaatteeltaan frekventistisen päättelyn suurimman uskottavuuden estimointia, jossa tarkoituksena on löytää estimaatti $\hat{\theta}$, joka maksimoi havaitun aineiston jakaumasta johdetun uskottavuusfunktion. Luvussa 2.6.2 esitelty pienimmän neliösumman menetelmä lineaariselle regressiolle vastaa suurimman uskottavuuden estimointia, kun virhetermit ovat normaalisti jakautuneita vakiovarianssilla. [7]

4.3 Bayesiläinen lineaarinen regressio ja regularisointi

Bayesiläisessä lineaarisessa regressiossa posteriorijakauma muodostetaan soveltamalla uskottavuusfunktiona normaalijakauman $\mathrm{N}(\mu, \sigma^2)$ tiheysfunktiota

$$p(y \mid \mathbf{x}, \boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2\right],$$

jossa μ (ja mahdollisesti myös σ) määräytyy selittävien muuttujien havainnosta $\mathbf{x} = (x_1, \ldots, x_p)$ ja tuntemattomista muuttujista $\boldsymbol{\theta}$. Havaitun aineiston muodostavat havaintomatriisi $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ ja vastevektori $\mathbf{y} = (y_1, \ldots, y_n)$. Tuntemattomia parametreja ovat regressiokertoimet $\beta_1, \ldots, \beta_p$. Käyttämällä tätä notaatiota posteriorijakaumaksi (9) saadaan [4]

$$p(\boldsymbol{\beta} \mid \mathbf{X}, \mathbf{y}) \propto p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\beta})\, p(\boldsymbol{\beta}).$$

Oletetaan, että vastemuuttujan $\mathbf{y}$ ehdollinen odotusarvo selittävillä muuttujilla $\mathbf{X}$ saadaan lineaarisena funktiona selittävistä muuttujista $\mathbf{X}$, eli

$$E(y_i \mid \mathbf{x}_i, \boldsymbol{\beta}) = \beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i.$$

Oletetaan lisäksi, että virhetermit ε ovat normaalijakautuneita keskiarvolla nolla ja vakiovarianssilla σ^2, eli $\varepsilon \sim \mathrm{N}(0, \sigma^2)$, ja että virhetermit ovat toisistaan riippumattomia. Näin ollen havaintojen uskottavuusfunktioksi saadaan [3]

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\beta}) &= \prod_{i=1}^{n} p(y_i \mid x_{i1}, \ldots, x_{ip}; \beta_0, \ldots, \beta_p) \\ &\propto \exp\left[-\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{n} x_{ij}\beta_j\right)^2\right] \\ &= \exp\left(-\frac{1}{2\sigma^2}\ \mathrm{RSS}\right). \end{aligned} \tag{10}$$

Jos posteriorijakauman muodostamisessa halutaan käyttää konjugaattiprioria, priorijakauman valinta riippuu siitä, mitkä mallin parametreista ovat tunnettuja. Jos esimerkiksi mallin varianssi on tunnettu, mutta keskiarvo ei, konjugaattipriorina voidaan käyttää normaalijakaumaa. Jos taas mallin varianssia ei tunneta, mutta keskiarvo tunnetaan, priorina voi käyttää käänteistä gammajakaumaa. Jos sekä varianssi että keskiarvo ovat tuntemattomia, priorina voidaan käyttää normaalia käänteistä gammajakaumaa. [8]

Normaalille uskottavuudelle voidaan myös johtaa epäinformatiivinen priori käyttämällä Jeffreysin prioria. Jeffreysin priori regressiokertoimille $\beta_1, \ldots, \beta_p$ ja variansille σ^2 on muotoa [9]

$$p\left(\boldsymbol{\beta}, \sigma^2\right) \propto \left(\sigma^2\right)^{-(p+2)/4}.$$

Bayesiläisessä lineaarisessa regressiossa priorijakauman $p(\boldsymbol{\beta})$ valinnalla voidaan vaikuttaa siihen, mitä arvoja regressiokertoimet $\beta_1, \ldots, \beta_p$ voivat saada ja millä todennäköisyydellä. Priorijakauman valinnalla voidaan siis tehdä myös mallin regularisointia: valitsemalla priorijakauma(t) siten, että se on pitkähäntäinen ja saavuttaa

huippunsa lähellä nollaa, posteriorijakauma muodostetaan sillä oletuksella, että jokainen selittävä muuttuja X_j ei todennäköisesti ole vaikutukseltaan merkittävä. Jos muuttuja kuitenkin saa nollasta poikkeavan regressiokertoimen, sen vaikutus voi olla hyvinkin merkittävä. [4]

Regularisoivan priorijakauman valinta ei ole yksiselitteistä. Priorijakauman sijainti ja skaala vaikuttavat regularisoinnin voimakkuuteen: mitä keskittyneempi priorijakauma on, sitä enemmän regularisointia se toteuttaa. Myös jakauman analyyttinen muoto vaikuttaa regularisointiin: esimerkiksi normaalijakauma skaalaa kaikkia regressiokertoimien estimaatteja kohti priorin keskiarvoa yhtä voimakkaasti suhteessa toisiin estimaatteihin, kun taas Laplace-jakauma siirtää estimaattien sijaintia yhtä suurella erotuksella. Myös se, kuinka posteriorijakaumaa tulkitaan voi vaikuttaa regularisointiin — käyttämällä täyden posteriorin sijaan posteriorin moodia tuloksessa on vähemmän varianssia, ja se on siten enemmän regularisoitunut. [4]

4.4 Bayesiläinen harjaregressio ja lasso

Harjaregressiota vastaava bayesiläinen menetelmä on muodostaa MAP-estimaatti siten, että regressiokertoimille $\beta_1, \ldots, \beta_p$ valitaan priorijakaumaksi normaalijakauma keskiarvolla nolla ja jollain varianssilla τ^2, eli $\boldsymbol{\beta} \sim \mathrm{N}(0, \tau^2)$ [2][6]. Kun tämä yhdistetään uskottavuusfunktioon (10), posteriorijakaumaksi saadaan

$$\begin{aligned}
p(\boldsymbol{\beta} \mid \mathbf{X}, \mathbf{y}) &\propto \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{n} x_{ij}\beta_j\right)^2\right] \times \exp\left(-\frac{1}{2\tau^2}\sum_{j=1}^{p}\beta_j^2\right) \\
&= \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{n} x_{ij}\beta_j\right)^2 - \frac{1}{2\tau^2}\sum_{j=1}^{p}\beta_j^2\right] \\
&= \exp\left\{-\frac{1}{2\sigma^2}\left[\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{n} x_{ij}\beta_j\right)^2 + \frac{\sigma^2}{\tau^2}\sum_{j=1}^{p}\beta_j^2\right]\right\} \\
&= \exp\left[-\frac{1}{2\sigma^2}\left(\mathrm{RSS} + \frac{\sigma^2}{\tau^2}\sum_{j=1}^{p}\beta_j^2\right)\right].
\end{aligned}$$

Tämä vastaa harjaregressiota (6), jossa säätöparametrina on $\lambda = \frac{\sigma^2}{\tau^2}$. Priorijakauma voidaankin kirjoittaa myös muodossa $\boldsymbol{\beta} \sim \mathrm{N}\left(0, \frac{\sigma^2}{\lambda}\right)$.

Harjaregression tavoin myös lasso voidaan tulkita bayesiläisittäin määrittelemällä MAP-estimaatti posteriorijakaumasta, jonka priorijakaumana käytetään Laplace-jakaumaa lokaatiolla nolla ja skaalalla b, eli $\boldsymbol{\beta} \sim \mathrm{Laplace}(0, b)$ [2][4]. Laplace-jakauman $\mathrm{Laplace}(\mu, b)$ tiheysfunktio on

$$p(y \mid \mathbf{x}, \boldsymbol{\theta}) = \frac{1}{2b}\exp\left(-\frac{|y - \mu|}{b}\right),$$

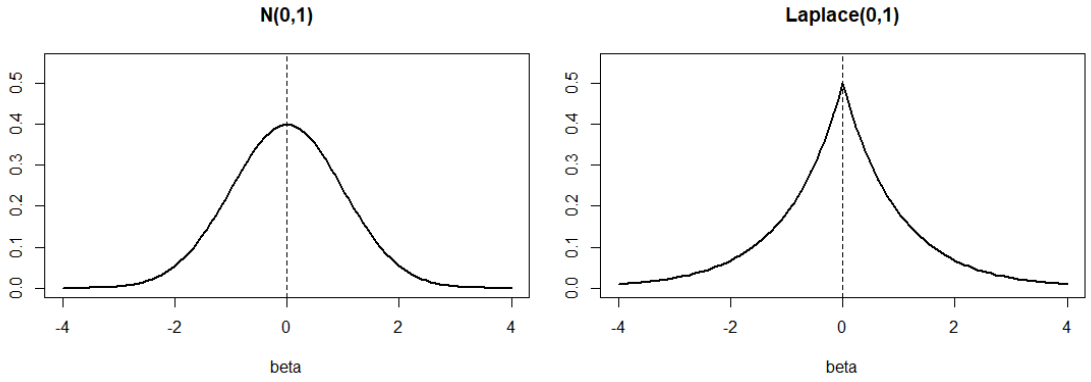
jossa skaalaparametri $b > 0$. Posteriorijakaumaksi saadaan tällöin

$$
\begin{aligned}
p(\boldsymbol{\beta} \mid \mathbf{X}, \mathbf{y}) &\propto \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{n} x_{ij}\beta_j\right)^2\right] \times \exp\left(-\frac{1}{b}\sum_{j=1}^{p}|\beta_j|\right) \\
&= \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{n} x_{ij}\beta_j\right)^2 - \frac{1}{b}\sum_{j=1}^{p}|\beta_j|\right] \\
&= \exp\left\{-\frac{1}{2\sigma^2}\left[\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{n} x_{ij}\beta_j\right)^2 + \frac{2\sigma^2}{b}\sum_{j=1}^{p}|\beta_j|\right]\right\} \\
&= \exp\left[-\frac{1}{2\sigma^2}\left(\mathrm{RSS} + \frac{2\sigma^2}{b}\sum_{j=1}^{p}|\beta_j|\right)\right],
\end{aligned}
$$

joka vastaa lassoregressiota (7), jossa säätöparametrina on $\lambda = \frac{2\sigma^2}{b}$. Priorijakauma voidaan tällöin kirjoittaa myös $\boldsymbol{\beta} \sim \text{Laplace}\left(0, \frac{2\sigma^2}{\lambda}\right)$.

Kuten luvussa 3.1 todettiin, harjaregressio kutistaa regressiomallin kertoimia $\beta_1, \ldots, \beta_p$ sekä kohti nollaa että toisiaan — yksikään kerroin ei kuitenkaan estimoidu tasan nollaksi, ja harjaregressio säilyttääkin mallissa kaikki mallin selittävät muuttujat. Lasso taas todennäköisesti kutistaa osan kertoimista tismalleen nollaksi, jolloin osa mallin selittävistä muuttujista jää käytännössä kokonaan pois mallista. Näitä havaintoja tukee myös regularisointimenetelmien bayesiläinen tulkinta.

Kuvassa 3 on kuvattu nollan ympärille keskitetyt normaalijakauma ja Laplace-jakauma. Normaalijakauma, jota käytetään bayesiläisen harjaregression priorina, saavuttaa huippunsa nollan ympärillä, mutta huippu ei ole kuitenkaan läheskään niin terävä kuin Laplace-jakaumalla, jota käytetään bayesiläisessä lassossa. Tämä selittää osaltaan sitä, miksi bayesiläinen lasso olettaa herkästi, että moni regressiokerroin on tismalleen nolla, kun taas harjaregressio olettaa, että kertoimet ovat tasaisesti jakautuneita nollan ympärillä. [2]



Kuva 3: Vasemmalla normaalijakauma keskiarvolla nolla, oikealla Laplace-jakauma lokaatiolla nolla.

5 Soveltava esimerkki

Tässä luvussa sovelletaan työssä esiteltyjä menetelmiä esimerkkiaineistoon. Sovelluksen tarkoituksena on esitellä työssä esiteltyjen menetelmien käyttöä käytännössä ja vahvistaa työn keskeinen sisältö: harjaregressio ja lasso tuottavat yleistettävämpiä ja siten parempia ennusteita verrattuna täyteen lineaariseen malliin, sekä harjaregression ja lasson tuottamat estimaatit regressiokertoimille vastaavat bayesiläisiä MAP-estimaatteja, kun niiden johtamisessa käytetään normaali- ja Laplace-prioreita. Soveltava esimerkki toteutettiin R- [10] ja Stan-ohjelmointikielillä [11].

5.1 Aineisto

Työn aineisto koostuu fyysisesti aktiivisten henkilöiden kehonmitoista. Havaintoja on yhteensä 507. Henkilöiltä mitattiin yhdeksän eri luuston halkaisijamittaa ja kaksitoista eri kehonosien ympärysmittaa. Lisäksi aineisto sisältää henkilöiden iän, painon, pituuden ja sukupuolen — aineistossa on siis yhteensä 25 muuttujaa. Kaikki muuttujat ovat numeerisia; sukupuolimuuttuja on binäärinen. Aineiston muuttujat on esitelty tarkemmin taulussa 1. Aineisto koostuu pääosassa 20–30-vuotiaista, ja se kerättiin Yhdysvaltojen Kalifornian osavaltiossa San Josén osavaltionyliopistossa, Yhdysvaltain laivaston tohtorikoulussa sekä muutamissa kuntosaleissa. Aineiston tarkoituksena oli tarjota tilastotieteen opiskelijoille aineisto, jota käyttää data-analyysin harjoittelussa. Aineisto ei ole satunnainen otos hyvin määritellystä populaatiosta, joten sen perusteella ei voida tehdä todellisuudessa yleistettäviä päätelmiä, mutta se soveltuu kuitenkin erilaisten tilastollisten menetelmien havainnollistamiseen. Aineisto julkaistiin alun perin osana artikkelia *Exploring Relationships in Body Dimensions* julkaisussa *Journal of Statistics Education* [12], ja työssä käytettiin sen implementaatiota R-paketissa *openintro* [13].

Aineistoon sovellettiin työssä esiteltyjä tilastollisia menetelmiä siten, että muuttuja *wgt* eli henkilöiden paino oli mallien vastemuuttujana ja kaikki loput 24 muuttujaa mallien selittävinä muuttujina. Jotta aineistolla saatiin paremmin simuloitua tilannetta, jossa ylisovittumista voisi tapahtua, aineiston 507 havainnosta valittiin satunnainen 100 havainnon otos. Eri menetelmien malleja sovitettaessa ja parametreja säädettäessä opetusaineistona käytettiin 80 %:a havainnoista ja testiaineistona 20 %:a havainnoista. Lopulliset parhaat mallit sovitettiin koko 100 havainnon otoksella, ja niiden testaamista varten aiemmin käyttämättömien 407 havainnon joukosta otettiin vielä täysin erillinen 50 havainnon satunnaisotos ns. ”lopulliseksi” testiaineistoksi. Opetusaineiston muuttujat normalisoitiin keskiarvolle nolla ja varianssille yksi, ja opetusaineistojen alkuperäistä keskiarvoa ja varianssia käytettiin myös testiaineistojen normalisointiparametreina.

5.2 Menetelmät

Aineistoon sovellettiin erilaisia regressiomenetelmiä käyttäen eri R-pakettien valmiita implementaatioita. Aineistoon sovitettiin täysi lineaarinen malli käyttäen R:n natiivia pakettia *stats* [10]. Frekventistiset harjaregressio- ja lassomallit sovitettiin käyttäen pakettia *glmnet* [14]. Harjaregressiomalli sovitettiin myös paketilla

Muuttujan nimi	Selite
bia_di	Olkalisäkkeiden etäisyys, ”hartioiden leveys” (cm)
bii_di	Suoliluun uloimpien kohtien etäisyys, ”lantion leveys” (cm)
bit_di	Isojen sarvennoisten etäisyys (cm)
che_de	Rinnan syvyys selkärangan ja rintalastan välillä (cm)
che_di	Rinnan leveys (cm)
elb_di	Kyynärpäiden leveys, molemmat kyynärpäät yhteensä (cm)
wri_di	Ranteiden leveys, molemmat ranteet yhteensä (cm)
kne_di	Polvien leveys, molemmat polvet yhteensä (cm)
ank_di	Nilkkojen leveys, molemmat nilkat yhteensä (cm)
sho_gi	Hartioiden ympärysmitta olkapäälihasten päältä (cm)
che_gi	Rinnan ympärysmitta (cm)
wai_gi	Vyötärön ympärysmitta ylävartalon kapeimmasta kohdasta (cm)
nav_gi	Vatsan ympärysmitta navan kohdalta (cm)
hip_gi	Lantion ympärysmitta isojen sarvennoisten tasolta (cm)
thi_gi	Reiden ympärysmitta, oikean ja vasemman keskiarvo (cm)
bic_gi	Hauiksen ympärysmitta, oikean ja vasemman keskiarvo (cm)
for_gi	Kyynärvarren ympärysmitta, oikean ja vasemman keskiarvo (cm)
kne_gi	Polven ympärysmitta, oikean ja vasemman keskiarvo (cm)
cal_gi	Säären ympärysmitta, oikean ja vasemman keskiarvo (cm)
ank_gi	Nilkan ympärysmitta, oikean ja vasemman keskiarvo (cm)
wri_gi	Ranteen ympärysmitta, oikean ja vasemman keskiarvo (cm)
age	Ikä (a)
wgt	Paino (kg)
hgt	Pituus (cm)
sex	Sukupuoli (mies = 1, nainen = 0)

Taulukko 1: Aineiston muuttujat ja niiden selitteet.

MASS [15] ja lassomalli paketilla *lars* [16]. Bayesiläinen analyysi toteutettiin Stan-ohjelmointikielellä R:n käyttöliittymässä käyttäen pakettia *rstan* [11]. Sovelluksessa käytettiin myös satunnaisia apufunktioita paketeista *caret* [17], *miscTools* [18] ja *TeachingDemos* [19].

Soveltavan esimerkin tavoitteena oli havainnollistaa työn kahta keskeistä väitettä: lineaarisen regression regularisointimenetelmät tuottavat parempia ennusteita kuin ylisovittunut täysi lineaarinen malli, ja harjaregressiota sekä lassoa vastaavat regressiokertoimet saadaan bayesiläisittäin tuotettua MAP-estimoinnilla. Käytännössä tämä tarkoitti, että kaikilla työssä esitellyillä menetelmillä pyrittiin toteuttamaan kaksi asiaa: johtaa estimaatit regressiokertoimille ja arvioida estimoidun mallin ennustevirhe. Työssä käytetyt bayesiläiset menetelmät riippuvat jossain määrin satunnaisuudesta — jotta tulokset olivat toistettavissa, menetelmät alustettiin tietyllä siemenluvulla. Yksinkertaisuuden vuoksi kaikki työssä esitellyt luvut raportoidaan kolmen desimaalin tarkkuudella.

Paketti *glmnet* vaikuttaa internetin hakutulosten perusteella hyvin yleiseltä tavalta sovittaa harjaregressio- ja lassomalleja R-ohjelmointikielellä. Siinä ratkaistava minimointiongelma eroaa kuitenkin hieman tässä työssä esitellystä ongelmasta: jäännösneliösumman minimoinnin sijaan *glmnet* minimoi jäännösneliösummaa jaettuna mallin opetuksessa käytettävän aineiston havaintojen lukumäärällä. Toisin sanoen, jotta *glmnet*-paketin käyttämä säätöparametri vastaisi tässäkin työssä käytettyä harjaregression ja lasson esitystapaa, se pitää kertoa havaintojen lukumäärällä. Tämän eroavaisuuden huomioimiseksi sovelluksessa käytetään *glmnet*-paketin säätöparametrista merkintää λ^*. Muuten säätöparametria merkitään samoin kuten muussakin työssä eli λ:lla, jolle siis pätee $\lambda = n\lambda^*$. Vertailun vuoksi harjaregressio- ja lassomallit päätettiin sovittaa myös vaihtoehtoisilla paketeilla *MASS* ja *lars*, joiden minimointiongelmat vastaavat tässä työssä käytettyä esitystapaa.

Mallien parametrien valinnoissa ja eri mallien ennustevirheiden vertailussa keskenään hyödynnettiin 5-kertaista ristiinvalidointia: aineisto jaettiin satunnaisesti viiteen osaan, joita yhtä kerrallaan käytettiin mallin testiaineistona ja muita neljää opetusaineistona. Kaikissa parametrien ja mallien arvioinneissa käytettiin samoja aineiston osituksia sekä harjaregression ja lasson kohdalla samoja mahdollisia säätöparametrin λ arvoja: näitä arvoja oli yhteensä 51 väliltä $[0.001, 100]$. Ennustevirheen mittana käytettiin jäännösvirhehajontaa (RMSE, *root mean squared error*), joka lasketaan ottamalla neliöjuuri keskineliövirheestä MSE (1). Jos jäännösvirhehajonta saa arvoksi nolla, mallin tuottamat ennusteet vastaavat täydellisesti uusien havaintojen todellisia arvoja. Toisin sanoen, mitä lähempänä jäännösvirhehajonta on nollaa, sitä parempia ennusteita malli tuottaa. Ristiinvalidoinnissa eri osituksilla saaduista jäännösvirhehajonnoista otettiin lopuksi vielä keskiarvo, eli ristiinvalidointien tuloksena oli kunkin mallin keskimääräinen jäännösvirhehajonta.

Kun vertailtiin, tuottavatko regularisointimenetelmät parempia ennusteita täyteen lineaariseen malliin verrattuna, harjaregression ja lasson arvioinnissa hyödynnettiin sisäkkäistä ristiinvalidointia (*nested cross-validation*). Sisäkkäinen ristiinvalidointi soveltuu tilanteisiin, joissa halutaan valita paras mallinnusmenetelmä sellaisten menetelmien joukosta, jotka itsessään riippuvat myös parhaiden mallin hyperparametrien valinnasta, kuten harjaregressio ja lasso. Sisäkkäisessä ristiinvalidoinnissa hyödynnetään kahta sisäkkäistä ristiinvalidointisilmukkaa: sisempi silmukka arvioi

mallia eri hyperparametreilla, ja ulompi silmukka arvioi sisemmän silmukan valitsemia malleja. Näin mallien lopullista suoriutumiskykyä pystytään arvioimaan täysin erillisellä testiaineistolla. Jos malleja vertailtaisiin vain yhden ristiinvalidointisilmukan perusteella, samoja testiaineistoja käytettäisiin sekä hyperparametrien valitsemiseen että lopullisen ennustevirheen arviointiin, mikä voi johtaa ylisovittumiseen ja todellista positiivisempiin tuloksiin.

Sovelluksessa käytetiin 5-kertaista ristiinvalidointia sekä sisemmässä että ulommassa silmukassa: 100 havainnon otos jaettiin siis ensin ulommassa silmukassa viiteen osaan, joita yhtä kerrallaan käytettiin ulomman silmukan testiaineistona ja loppuja neljää osaa opetusaineistona. Tämä opetusaineisto jaettiin uudestaan sisemmässä silmukassa viiteen osaan, joista yhtä käytettiin mallin hyperparametrien arvioinnin validointiaineistona, ja loppuja neljää osaa opetusaineistona. Sisempi silmukka valitsi siis viisi parasta hyperparametrin arvoa, joilla sovitettuja malleja arvioitiin vielä ulommassa silmukassa täysin erillisellä testiaineistolla. Sisäkkäisen ristiinvalidoinnin antama tulos oli näiden viiden mallin keskimääräinen ennustevirhe. Koska täyden lineaarisen mallin sovittaminen ei riipu minkään hyperparametrin valinnasta, sen arvioinnissa käytettiin vain yhtä 5-kertaista ristiinvalidointisilmukkaa. Sisäkkäisellä ristiinvalidoinnilla arvioitiin vain frekventististen harjaregressio- ja lassomenetelmien suoriutumista täyteen lineaariseen malliin verrattuna.

Ennustevirheiden vertailun lisäksi toisena pääkohtana sovelluksessa oli vertailla keskenään frekventististen regularisointimenetelmien ja bayesiläisen MAP-estimoinnin tuottamia regressiokertoimia — teoriassa molempien lähestymistapojen pitäisi tuottaa samat regressiokertoimet. Vertailua varten sovitettiin koko 100 havainnon otosta hyödyntäen sekä täysi lineaarinen malli että eri harjaregressioja lassomallit. Frekventististen mallien säätöparametrit valittiin jälleen 5-kertaisella ristiinvalidoinnilla, ja näitä samoja säätöparametrin arvoja käytettiin vertailtavien MAP-estimaattien muodostamisessa. Regressiokertoimien vertailun lisäksi näiden lopullisten, ns. ”parhaiden” mallien ennustevirhettä arvioitiin vielä mielenkiinnosta täysin erilliseen 50 havainnon satunnaisotokseen alkuperäisestä aineistosta.

Bayesiläistä analyysiä varten muodostettiin sekä harjaregressiota että lassoa vastaavat mallit Stan-ohjelmointikielellä. Mallien käyttämien priorijakaumien tarkemmat määritelmät on johdettu luvussa 4.4. Harjaregressiota vastaava malli oli muotoa

$$\begin{aligned} \boldsymbol{\beta} &\sim \mathrm{N}\left(0, \sqrt{\frac{\sigma_e^2}{\lambda}}\right), \\ \mathbf{y} &\sim \mathrm{N}\left(\mathbf{X}\boldsymbol{\beta}, \sigma_e\right), \end{aligned} \tag{11}$$

jossa sekä priorijakaumana että vastemuuttujan jakaumana on normaalijakauma, jonka parametreina on keskiarvo ja keskihajonta. Mallissa esiintyvä σ_e on lineaarisen mallin residuaalien keskihajonta. Lassoa vastaava malli oli muotoa

$$\begin{aligned} \boldsymbol{\beta} &\sim \mathrm{Laplace}\left(0, \frac{2\sigma_e^2}{\lambda}\right), \\ \mathbf{y} &\sim \mathrm{N}\left(\mathbf{X}\boldsymbol{\beta}, \sigma_e\right), \end{aligned} \tag{12}$$

jossa vastemuuttujan jakaumana on harjaregressiota vastaavasti normaalijakauma, mutta priorijakaumana on Laplace-jakauma, jonka parametreja ovat lokaatio ja skaa-

la. Mallien säätöparametrit λ asetettiin samoiksi, jotka valittiin kunkin frekventistisen mallinnusmenetelmän mukaan parhaaksi arvoksi.

Vaikka MAP-estimointi onkin tämän työn kontekstissa erityisenä mielenkiinnon kohteena, MAP-estimointi yleisesti ei välttämättä ole kaikkien luonnollisin tapa johtaa estimaatteja bayesiläisittäin. Bayesiläisempi lähestymistapa olisi estimoida regressiokertoimet posteriorijakaumasta johdetusta otoksesta joko keskiarvoina tai mediaaneina. Myös keskiarvot ja mediaanit kutistavat regressiokertoimia kohti nollaa, mutta ei kuitenkaan tasan nollaksi, mikä vaikuttaa erityisesti lassolla mallintamiseen. Posteriorin keskiarvoilla ja mediaaneilla on kuitenkin etuna esimerkiksi se, että niille saadaan johdettua luottovälit, jotka voivat auttaa mahdollisessa muuttujanvalinnassa. Sovelluksessa verrataankin siis MAP-estimaatteja vielä mielenkiinnosta posteriorimediaaneihin, joita on käytetty erityisesti bayesiläisessä lassomallintamisessa myös muussa kirjallisuudessa. [20]

5.3 Tulokset

5.3.1 Ennustevirheiden vertailu

Aineistoon sovitettiin ensin lineaarinen malli siten, että vastemuuttujaa *wgt* eli henkilöiden painoa pyrittiin selittämään kaikilla muilla aineiston 24 muuttujalla. Lineaarinen malli soveltui aineistoon melko hyvin: sen keskimääräinen ennustevirhe jäännösvirhehajonnalla mitattuna oli 0.200, ja ylipäätään mallin ennustevirhe vaihteli eri ristiinvalidoinnin kierroksilla välillä $[0.146, 0.233]$. Vähintään 90 %:n luottamustasolla tilastollisesti merkitseviä muuttujia malleissa oli osituksesta riippuen 6–10, joista neljä oli merkitseviä kaikilla osituksilla: *wai_gi* eli vyötärön ympärysmitta, *hip_gi* eli lantion ympärysmitta, *age* eli henkilöiden ikä, ja *hgt* eli henkilöiden pituus. Läheskään kaikki täyden lineaarisen mallin muuttujat eivät siis vaikuta olevan kovinkaan tärkeitä mallissa, joten voisi olla perusteltua ajatella, että yksinkertaistamalla mallia regularisointimenetelmillä voitaisiin saavuttaa vielä parempi ennustetarkkuus.

Aineistoon sovitettiin seuraavaksi harjaregressiomalli *glmnet*-paketilla ja *MASS*-paketilla, ja niiden keskimääräistä ennustevirhettä testattiin sisäkkäisellä ristiinvalidoinnilla. Kummallakin menetelmällä keskimääräiseksi jäännösvirhehajonnaksi saatiin 0.189, eli harjaregressiolla saavutettiin jonkin verran parempi ennustetarkkuus kuin täydellä lineaarisella mallilla. *glmnet*-menetelmällä ennustevirhe vaihteli välillä $[0.135, 0.223]$ ja *MASS*-menetelmällä välillä $[0.134, 0.222]$.

Aineistoon sovitetiin myös lassomalli kahdella eri implementaatiolla: paketeilla *glmnet* ja *lars*. Sekä *glmnet*- että *lars*-pakettien mallinnusmenetelmien keskimääräinen ennustevirhe sisäkkäisellä ristiinvalidoinnilla testattuna oli 0.193. Ennustevirheet vaihtelivat menetelmillä väleillä $[0.136, 0.244]$ (*glmnet*) ja $[0.138, 0.242]$ (*lars*). Kaiken kaikkian molemmilla lassomenetelmillä saatiin hieman pienempi ennustevirhe täyteen lineaariseen malliin verrattuna, mutta ennustevirheet olivat kuitenkin keskimäärin huonompia kuin harjaregressiomenetelmillä.

Kaiken kaikkiaan lineaarisen regression tuloksia aineistolla saatiin parannettua, kun täyden lineaarisen mallin sijaan käytettiin harjaregressiota tai lassoa aineiston mallintamiseen. Parhaat ennustetarkkuudet saatiin harjaregressiomenetelmillä. Erot keskimääräisessä ennustevirheessä eivät kuitenkaan olleet suuria, ja joillain ristiin-

validoinnin kierroksilla täydellä lineaarisella mallilla saatiin jopa parempia tuloksia kuin regularisointimenetelmillä. Tarkemmat ennustevirheet jäännösvirhehajonnalla mitattuna on esitelty taulukossa 2.

	Lineaarinen malli	*glmnet*-harjaregressio	*MASS*-harjaregressio	*glmnet*-lasso	*lars*-lasso
$k = 1$	0.195	0.183	0.186	0.200	0.198
$k = 2$	0.212	0.179	0.179	0.174	0.175
$k = 3$	0.146	0.135	0.134	0.136	0.138
$k = 4$	0.204	0.213	0.213	0.192	0.197
$k = 5$	0.233	0.223	0.222	0.244	0.242
Ennuste-virhe	0.200	0.189	0.189	0.193	0.193

Taulukko 2: Jäännösvirhehajonnat ristiinvalidoinnin tai sisäkkäisen ristiinvalidoinnin ulomman silmukan eri kierroksilla k. Viimeisenä koko arvioinnin lopputulos eli keskimääräinen jäännösvirhehajonta.

5.3.2 Regressiokertoimien vertailu

Regressiokertoimien vertailua varten koko 100 havainnon otosta käytettiin opettamaan vertailtavat mallit. Harjaregressiomallien vertailua varten sovitettiin ensin frekventistiset mallit *glmnet*- ja *MASS*-paketeilla. Kummankin mallinnusmenetelmän säätöparametrin arvo valittiin 5-kertaisella ristiinvalidoinnilla: *glmnet*-mallin säätöparametriksi valittiin $\lambda^* = 0.040$ (0.03981...) ja *MASS*-mallin säätöparametriksi valittiin $\lambda = 3.981$. Molempien mallien regressiokertoimia estimoitaessa käytettiin siis käytännössä samaa säätöparametria, sillä $n\lambda^* = 100 \times 0.03981... = 3.981$. Siitä huolimatta menetelmät eivät tuottaneet tismalleen samoja regressiokertoimia, vaan niissä esiintyi pieniä, muutaman tuhannesosan suuruisia eroja.

Näitä malleja vastaavat MAP-estimaatit laskettiin käyttäen harjaregressiota vastaavaa Bayes-mallia (11): estimaatit laskettiin käyttäen sekä havaintojen määrällä skaalattua *glmnet*-mallin säätöparametria ja *MASS*-mallin säätöparametria sellaisenaan. Koska säätöparametrit olivat käytännössä samat, kumpikin MAP-estimointi tuotti odotetusti käytännössä samat regressiokertoimet. Ne eivät olleet tismalleen samat, vaan niissä alkoi näkyä pieniä eroavaisuuksia viidennen tai kuudennen merkitsevän numeron kohdalla. Samoilla kahdella Bayes-mallilla johdetut regressiokertoimet, jotka laskettiin posterioriotoksen mediaaneina, poikkesivat hieman enemmän keskenään, mutta niidenkin kohdalla erot olivat muutaman tuhannesosan kokoluokkaa. Kaikki estimoidut regressiokertoimet on esitelty tarkemmin taulukossa 3.

MAP-estimoidut regressiokertoimet harjaregressiolle olivat lähestulkoon samat, kuin niitä vastaavat kertoimet frekventistisillä *glmnet*- ja *MASS*-malleilla. Niissä oli jälleen tuhannesosien suuruisia eroja. *MASS*-mallin kertoimet vaikuttivat olevan aavistuksen lähempänä MAP-estimoituja kertoimia *glmnet*-malliin verrattuna:

MASS-mallin kertoimien erot MAP-estimaatteihin esiintyivät toisen tai kolmannen merkitsevän numeron kohdalla, mutta *glmnet*-malliin verrattuna eroavaisuuksia saattoi olla jo ensimmäisen merkitsevän numeron tarkkuudella. Tämä päti erityisesti niihin regressiokertoimiin, jotka kutistuivat kaikkein lähimmäksi nollaa.

MAP-estimoinnilla saatiin käytännössä siis *melkein* samat estimaatit regressiokertoimille kuin frekventistisillä harjaregressiomenetelmillä. Pieniä eroavaisuuksia ilmeni myös eri R-paketteihin implementoiduilla harjaregression mallintamismenetelmillä. Kaikilla menetelmillä — myös posteriorimediaaneilla — saatiin kuitenkin lähestulkoon samat estimaatit regressiokertoimille. Mallien ennustetarkkuutta mitattaessa näillä pienillä eroilla regressiokertoimissa ei ollut suurta merkitystä: kun malleilla ennustettiin täysin uutta 50 havainnon aineistoa, niiden kaikkien ennustevirhe oli samaa tasoa, 0.165. Samalla aineistolla opetetun täyden lineaarisen mallin ennustevirhe oli aavistuksen suurempi, 0.175.

Harjaregressiomallien tavoin aineistoon sovitettiin myös lassomallit. Lassomallien säätöparametreiksi saatiin *glmnet*-paketilla $\lambda^* = 0.008$ (0.00794...) ja *lars*-paketilla $\lambda = 0.631$. Harjaregressiomenetelmistä poiketen mallien valitsemissa säätöparametreissa oli hieman eroavaisuutta. Molemmissa malleissa kutistui tasan nollaksi samat kahdeksan selittävää muuttujaa, ja malliin jääneiden muuttujien regressiokertoimet ovat melko lähellä toisiaan. *Glmnet*-lassomallilla saatiin siitä huolimatta kuitenkin aavistuksen parempi ennustetarkkuus uudella aineistolla: *glmnet*-lassomallin jäännösvirhehajonta oli 0.170, kun taas *lars*-mallilla se oli 0.171.

Glmnet- ja *lars*-malleilla saadut lassomallien regressiokertoimet eroavat kuitenkin niitä vastaavilla malleilla saaduista MAP-estimaateista enemmän kuin harjaregressiomallien kertoimet erosivat niitä vastaavista MAP-estimaateista. Taulukosta 4 nähdään, että frekventististen menetelmien ja MAP-estimoinnin tuottamat regressiokertoimet ovat karkeasti samaa kokoluokkaa, mutta niissä esiintyy paikoin jopa sadasosien suuruisia eroja. Suurimmalle osalle muuttujista, jotka jäivät frekventisistä malleista kokonaan pois, myös MAP-estimoitu regressiokerroin oli käytännössä nolla tai ainakin melko lähellä sitä, mutta joillekin niistä jäi jopa verrattain melko suuri regressiokerroin. Vaikka regressiokertoimissa olikin eroa, MAP-estimoinnilla saavutettiin sama ennustetarkkuus kuin niitä vastaavilla frekventistisillä lassomalleilla: MAP-estimoinnin ennustevirhe *glmnet*-mallin säätöparametrilla oli 0.170 ja *lars*-mallin säätöparametrilla 0.171.

Myös posteriorimediaaneilla saatiin karkeasti samansuuruisia regressiokertoimia kuin frekventistisellä estimoinnilla ja MAP-estimoinnilla. Erityisesti poikkeavuuksia oli niiden muuttujien kertoimissa, jotka kutistuivat *glmnet-* ja *lars*-malleissa tasan nollaksi. Posteriorimediaaneilla saavutettiin kuitenkin uudella aineistolla testattuna aavistuksen parempi ennustetarkkuus kuin muilla lassomalleilla: *glmnet*-mallin säätöparametrilla ennustevirhe oli 0.167 ja *lars*-mallin säätöparametrilla 0.168.

	glmnet: $\lambda^* = 0.040$			*MASS*: $\lambda = 3.981$		
	Harja-regressio	MAP	Posteriori-mediaani	Harja-regressio	MAP	Posteriori-mediaani
bia_di	-0.004	-0.005	-0.006	-0.005	-0.005	-0.004
bii_di	0.025	0.025	0.025	0.025	0.025	0.025
bit_di	-0.010	-0.010	-0.011	-0.010	-0.010	-0.011
che_de	0.078	0.076	0.076	0.076	0.076	0.076
che_di	0.045	0.043	0.043	0.043	0.043	0.042
elb_di	-0.014	-0.014	-0.013	-0.014	-0.014	-0.013
wri_di	0.009	0.010	0.010	0.010	0.010	0.010
kne_di	0.051	0.052	0.052	0.052	0.052	0.052
ank_di	0.017	0.018	0.018	0.018	0.018	0.017
sho_gi	0.092	0.095	0.095	0.094	0.095	0.095
che_gi	0.085	0.088	0.087	0.087	0.088	0.089
wai_gi	0.223	0.224	0.224	0.223	0.224	0.224
nav_gi	0.034	0.034	0.034	0.033	0.034	0.033
hip_gi	0.173	0.172	0.174	0.172	0.172	0.174
thi_gi	0.063	0.062	0.060	0.061	0.062	0.061
bic_gi	0.082	0.082	0.082	0.082	0.082	0.083
for_gi	0.102	0.100	0.102	0.100	0.100	0.099
kne_gi	0.066	0.066	0.066	0.065	0.066	0.065
cal_gi	0.082	0.081	0.082	0.081	0.081	0.081
ank_gi	-0.016	-0.015	-0.015	-0.015	-0.015	-0.015
wri_gi	-0.020	-0.021	-0.022	-0.021	-0.021	-0.021
age	-0.071	-0.072	-0.072	-0.072	-0.072	-0.071
hgt	0.130	0.129	0.130	0.129	0.129	0.129
sex	-0.005	-0.007	-0.007	-0.007	-0.007	-0.007
Ennuste-virhe	0.165					

Taulukko 3: Frekventististen harjaregressiomenetelmien tuottamat kertoimet sekä niitä vastaavat kertoimet bayesiläisillä menetelmillä. Ennustevirheen laskennassa käytettiin uutta satunnaisotosta alkuperäisestä aineistosta. Vastaava ennustevirhe täydellä lineaarisella mallilla oli 0.175.

	glmnet: $\lambda^* = 0.008$			*lars*: $\lambda = 0.631$		
	Lasso	MAP	Posteriori-mediaani	Lasso	MAP	Posteriori-mediaani
bia_di	.	<0.001	-0.007	.	-0.003	-0.008
bii_di	0.017	0.020	0.021	0.018	0.022	0.021
bit_di	.	-0.014	-0.017	.	-0.021	-0.022
che_de	0.065	0.069	0.066	0.066	0.069	0.067
che_di	0.032	0.033	0.033	0.031	0.032	0.031
elb_di	.	<0.001	-0.010	.	<0.001	-0.013
wri_di	.	<0.001	0.005	.	<0.001	0.005
kne_di	0.047	0.047	0.049	0.048	0.050	0.050
ank_di	0.003	0.011	0.019	0.004	0.016	0.024
sho_gi	0.086	0.088	0.090	0.086	0.096	0.093
che_gi	0.084	0.066	0.070	0.086	0.058	0.064
wai_gi	0.269	0.274	0.274	0.270	0.282	0.281
nav_gi	.	<0.001	0.010	.	<0.001	0.005
hip_gi	0.181	0.203	0.191	0.182	0.201	0.198
thi_gi	0.067	0.053	0.055	0.067	0.055	0.051
bic_gi	0.078	0.088	0.078	0.079	0.092	0.080
for_gi	0.083	0.092	0.109	0.080	0.088	0.112
kne_gi	0.063	0.064	0.064	0.063	0.065	0.064
cal_gi	0.068	0.071	0.072	0.068	0.070	0.073
ank_gi	.	-0.006	-0.011	.	-0.009	-0.015
wri_gi	.	-0.005	-0.014	.	-0.006	-0.016
age	-0.062	-0.073	-0.076	-0.066	-0.076	-0.079
hgt	0.123	0.130	0.137	0.123	0.134	0.142
sex	.	-0.011	-0.015	.	-0.016	-0.022
Ennuste-virhe	0.170		0.167	0.171		0.168

Taulukko 4: Frekventististen lassomenetelmien tuottamat kertoimet sekä niitä vastaavat kertoimet bayesiläisillä menetelmillä. Merkintä ” . ” tarkoittaa muuttujaa, joka kutistui mallista kokonaan pois. Ennustevirheen laskennassa käytettiin uutta satunnaisotosta alkuperäisestä aineistosta. Vastaava ennustevirhe täydellä lineaarisella mallilla oli 0.175.

5.4 Pohdinta

Työssä käytetyllä esimerkkiaineistolla pystyttiin näyttämään, että regularisointimenetelmillä voidaan saavuttaa parempi ennustetarkkuus kuin täydellä lineaarisella mallilla. Myös täysi lineaarinen malli soveltui käytettyyn aineistoon melko hyvin, mutta tuloksia saatiin jonkin verran parannettua kaikilla eri regularisointimenetelmillä. Luvussa 3.2 mainittiin, että harjaregressiolla saavutetaan tyypillisesti hieman parempi ennustetarkkuus kuin lassolla — näin kävi myös työn esimerkkiaineiston tapauksessa.

Kun aineiston perusteella valittujen parhaiden frekventististen mallien tuottamia regressiokertoimia vertailtiin niiden teoreettisiin MAP-estimoituihin vastineisiin, harjaregressiomenetelmillä saatiin lähes yhtenevät tulokset, mutta lassomenetelmillä eroa oli havaittavissa enemmän. Sekä frekventistiset harjaregressio- ja lassomallit sovitettiin kahdella eri R-paketin menetelmällä, ja mielenkiintoista oli, että myös niiden välillä esiintyi pieniä eroja estimoiduissa regressiokertoimissa. Tätä tutkittiin vielä lisää sovittamalla kaikilla työssä käytetyillä menetelmillä 51 eri mallia niillä 51:llä eri säätöparametrin λ arvolla, joita käytettiin sovittamaan eri mallit myös aikaisemmin työssä. Jokaisella eri säätöparametrin arvolla sovitettujen mallien regressiokertoimien eroa mitattiin jäännösvirhehajonnalla, joiden perusteella laskettiin keskimääräinen virhe eri mallinnusmenetelmien kertoimien välillä. Nämä tulokset on esitelty tarkemmin taulukossa 5.

Tämänkin tarkastelun perusteella harjaregressiomalleilla saatiin lähes yhtenevät tulokset. MAP-estimoitujen mallien ero *glmnet*- ja *MASS*-malleihin oli lähes olematonta, erityisesti *MASS*-mallien kertoimiin verrattuna. *glmnet*- ja *MASS*-pakettien tuottamien mallien välillä oli keskimäärin jopa enemmän eroa kuin niillä oli MAP-estimoituihin kertoimiin verrattuna. Harjaregression kertoimet saadaan helposti ratkaistua myös ns. käsin suorana matriisilaskuna. Nämä kertoimet olivat keskimäärin lähes tismalleen samat MAP-estimoitujen kertoimien kanssa (RMSE = 0.000012).

Lassomalleilla eri mallien kertoimissa oli kuitenkin enemmän eroavaisuutta. Kaiken kaikkiaan jäännösvirhehajonta, joka oli noin 0.025 kaikkien lassomenetelmien välillä ei ole kovinkaan suuri, mutta harjaregressiomenetelmien välisiin eroihin (noin 0.001) verrattuna se alkaa vaikuttaa merkittävältä.

Erot MAP-estimaattien sekä *glmnet*-, *MASS*- ja *lars*- pakettien tuottamien kertoimien välillä johtunevat ainakin osittain niiden mallintamisfunktioihin implementoiduista erilaisista asetuksista: erityisesti *glmnet*-paketti tarjoaa käyttäjilleen hyvin paljon erilaisia mahdollisuuksia muokata mallintamisfunktiota erilaisilla asetuksilla ja lisämääreillä. Toisaalta tämän ansiosta pakettien tarjoamia menetelmiä saadaan sovellettua monenlaisiin eri mallinnustilanteisiin ja malleja pystytään hienosäätämään monin eri tavoin. Toisaalta eri mahdollisuudet hienosäätää malleja monimutkaistavat niiden ytimessä olevaa minimointiongelmaa, jolloin lienee mahdollista, että niiden antamat tulokset alkavat poiketa siitä, mitä harjaregressio ja lasso yksinkertaisimmillaan tuottaisivat.

Erityisesti lasson kohdalla MAP-estimaattien eroa frekventistisesti tuotettuihin kertoimiin selittänee myös se, että bayesiläisittäin mallin priorijakaumana käytetään Laplace-jakaumaa. Kuten kuvasta 3 nähdään, Laplace-jakauma saavuttaa terävän piikin jakauman lokaation kohdalla. Tämä tarkoittaa, että jakauman tiheysfunktio ei ole derivoituva jakauman lokaatiossa, mikä aiheuttaa epävakautta erityisesti sel-

laisten regressiokertoimien estimoinnissa, joiden arvo olisi jakauman lokaatiossa tai ainakin hyvin lähellä sitä. Työn esimerkkiaineistoa mallintaessakin huomattiin, että erityisesti niiden muuttujien kertoimissa, jotka *glmnet*- ja *lars*-menetelmillä kutistuivat tasan nollaksi, esiintyi vaihtelevuutta kertoimia MAP-estimoitaessa.

Taulukossa 6 on vielä esitelty parhaat mallit, jotka jokainen työssä käytetty menetelmä valitsi koko työssä käytetyn 100 havainnon otoksella, ja niiden ennustetarkkuus uudella, 50 havainnon otoksella. Tätä varten myös MAP-estimoinnilla valittiin 5-kertaisella ristiinvalidoinnilla parhaat säätöparametrien λ arvot harjaregressio- ja lassomalleille. Kaikki harjaregressiomenetelmät valitsivat parhaaksi säätöparametrin arvoksi $\lambda = 3.981$ ja myös ennustetarkkuus oli kaikilla malleilla samaa tasoa, 0.165. Kaikki lassomallit valitsivat kuitenkin säätöparametrin arvoksi eri arvot, ja myös niiden ennustetarkkuuksissa oli eroa: *glmnet*- ja *lars*-mallien ennustevirhe oli noin 0.170, mutta MAP-estimoidulla mallilla saavutettiin sama ennustevirhe kuin harjaregressiomalleilla eli 0.165. Kaikkien regularisoivien mallien ennustetarkkuus oli parempi kuin täydellä lineaarisella mallilla, jonka ennustevirhe oli 0.175.

Harjaregressio	*glmnet* vs. MAP	*MASS* vs. MAP	*glmnet* vs. *MASS*
Virhe	0.001	<0.001	0.002
Lasso	***glmnet*** **vs. MAP**	***lars*** **vs. MAP**	***glmnet*** **vs. *lars***
Virhe	0.027	0.021	0.029

Taulukko 5: Erot eri mallinnusmenetelmien tuottamissa regressiokertoimissa jäännösvirhehajonnalla mitattuna. Virheet laskettiin keskiarvona 51:llä eri säätöparametrin λ arvolla sovitetuista malleista.

		Harjaregressio			Lasso		
	Täysi LM	*glmnet* $\lambda^* = 0.040$	*MASS* $\lambda = 3.981$	MAP $\lambda = 3.981$	*glmnet* $\lambda^* = 0.008$	*lars* $\lambda = 0.631$	MAP $\lambda = 1.995$
bia_di	-0.020	-0.004	-0.005	-0.005	.	.	<0.001
bii_di	0.029	0.025	0.025	0.025	0.017	0.018	0.016
bit_di	-0.044	-0.010	-0.010	-0.010	.	.	<0.001
che_de	0.072	0.078	0.076	0.076	0.065	0.066	0.062
che_di	0.034	0.045	0.043	0.043	0.032	0.031	0.029
elb_di	-0.038	-0.014	-0.014	-0.014	.	.	<0.001
wri_di	0.006	0.009	0.010	0.010	.	.	<0.001
kne_di	0.052	0.051	0.052	0.052	0.047	0.048	0.043
ank_di	0.046	0.017	0.018	0.018	0.003	0.004	<0.001
sho_gi	0.110	0.092	0.094	0.095	0.086	0.086	0.075
che_gi	0.031	0.085	0.087	0.088	0.084	0.086	0.119
wai_gi	0.312	0.223	0.223	0.224	0.269	0.270	0.238
nav_gi	-0.027	0.034	0.033	0.034	.	.	0.012
hip_gi	0.238	0.173	0.172	0.172	0.181	0.182	0.184
thi_gi	0.027	0.063	0.061	0.062	0.067	0.067	0.058
bic_gi	0.101	0.082	0.082	0.082	0.078	0.079	0.067
for_gi	0.162	0.102	0.100	0.100	0.083	0.080	0.089
kne_gi	0.066	0.066	0.065	0.066	0.063	0.063	0.063
cal_gi	0.070	0.082	0.081	0.081	0.068	0.068	0.080
ank_gi	-0.025	-0.016	-0.015	-0.015	.	.	<0.001
wri_gi	-0.041	-0.020	-0.021	-0.021	.	.	<0.001
age	-0.089	-0.071	-0.072	-0.072	-0.062	-0.066	-0.057
hgt	0.163	0.130	0.129	0.129	0.123	0.123	0.125
sex	-0.061	-0.005	-0.007	-0.007	.	.	<0.001
Ennuste-virhe	0.175	0.165	0.165	0.165	0.170	0.171	0.165

Taulukko 6: Kaikilla menetelmillä (pl. posteriorimediaanit) valitut parhaat mallit ja niiden ennustevirheet uudella aineistolla. Merkintä ” . ” tarkoittaa muuttujaa, joka kutistui mallista kokonaan pois.

Kirjallisuutta

[1] T. Hastie, R. Tibshirani ja J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Springer Series in Statistics), 2. painos. New York, NY: Springer, 2009, ISBN: 978-0-387-84857-0.

[2] G. James, D. Witten, T. Hastie ja R. Tibshirani, toim., *An Introduction to Statistical Learning: with Applications in R* (Springer Texts in Statistics). New York, NY: Springer, 2013, ISBN: 978-1-4614-7137-0.

[3] W. M. Bolstad ja J. M. Curran, *Introduction to Bayesian Statistics*, 3. painos. Hoboken, NJ: Wiley, 2017, ISBN: 978-1-118-09156-2.

[4] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari ja D. B. Rubin, *Bayesian Data Analysis* (Chapman & Hall/CRC Texts in Statistical Science), 3. painos. Boca Raton, FL: CRC Press, 2013, ISBN: 978-1-4398-4095-5.

[5] C. P. Robert, *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation* (Springer Texts in Statistics), 2. painos. New York, NY: Springer, 2007, ISBN: 978-0-387-71598-8.

[6] K. P. Murphy, *Machine Learning: A Probabilistic Perspective* (Adaptive Computation and Machine Learning Series). Cambridge, MA: MIT Press, 2012, ISBN: 978-0-262-01802-9.

[7] S. Theodoridis, *Machine learning: A Bayesian and Optimization Perspective*, 2. painos. Cambridge, MA: Academic Press, ISBN: 978-0-12-818803-3.

[8] K. Murphy, "Conjugate Bayesian analysis of the Gaussian distribution," 2007. url: `https://www.cs.ubc.ca/~murphyk/Papers/bayesGauss.pdf`.

[9] J. G. Ibrahim ja P. W. Laud, "On Bayesian Analysis of Generalized Linear Models Using Jeffreys's Prior," *Journal of the American Statistical Association*, vol. 86, nro 416, s. 981–986, 1991, ISSN: 0162-1459, 1537-274X. DOI: `10.1080/01621459.1991.10475141`.

[10] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2023. url: `https://www.R-project.org/`.

[11] Stan Development Team, *RStan: the R interface to Stan*, R package version 2.32.3, 2023. url: `https://mc-stan.org/`.

[12] G. Heinz, L. J. Peterson, R. W. Johnson ja C. J. Kerk, "Exploring Relationships in Body Dimensions," *Journal of Statistics Education*, vol. 11, nro 2, 2003, ISSN: 1069-1898. DOI: `10.1080/10691898.2003.11910711`.

[13] M. Çetinkaya-Rundel, D. Diez, A. Bray et al., *openintro: Data Sets and Supplemental Functions from 'OpenIntro' Textbooks and Labs*, R package version 2.4.0, 2022. url: `https://CRAN.R-project.org/package=openintro`.

[14] J. Friedman, R. Tibshirani ja T. Hastie, "Regularization Paths for Generalized Linear Models via Coordinate Descent," *Journal of Statistical Software*, vol. 33, nro 1, s. 1–22, 2010. DOI: `10.18637/jss.v033.i01`.

[15] W. N. Venables ja B. D. Ripley, *Modern Applied Statistics with S*, 4. painos. New York: Springer, 2002, ISBN 0-387-95457-0.

[16] T. Hastie ja B. Efron, *lars: Least Angle Regression, Lasso and Forward Stagewise*, R package version 1.3, 2022. url: https://CRAN.R-project.org/package=lars.

[17] Kuhn ja Max, "Building Predictive Models in R Using the caret Package," *Journal of Statistical Software*, vol. 28, nro 5, s. 1–26, 2008. DOI: 10.18637/jss.v028.i05.

[18] A. Henningsen ja O. Toomet, *miscTools: Miscellaneous Tools and Utilities*, R package version 0.6-28, 2023. url: https://CRAN.R-project.org/package=miscTools.

[19] G. Snow, *TeachingDemos: Demonstrations for Teaching and Learning*, R package version 2.13, 2024. url: https://CRAN.R-project.org/package=TeachingDemos.

[20] T. Park ja G. Casella, "The Bayesian Lasso," *Journal of the American Statistical Association*, vol. 103, nro 482, s. 681–686, 2008, ISSN: 0162-1459, 1537-274X. DOI: 10.1198/016214508000000337.

Liite A R-koodi

```
########## Kaytetyt paketit ##########

library(openintro) # Data bdims
library(caret) # Sekalaisia apufunktioita
library(miscTools) # Sekalaisia apufunktioita
library(TeachingDemos) # Sekalaisia apufunktioita
library(glmnet) # Harjaregressio ja lasso
library(MASS) # Harjaregressio
library(lars) # Lasso
library(rstan) # Bayes

########## Datan valmistelu ##########

# Alustus
set.seed(NULL)
seed_luku <- char2seed("Emilia", set = FALSE)
set.seed(seed_luku)

# Luetaaan data
set.seed(NULL)
set.seed(seed_luku)
data_i <- sample(nrow(bdims), 100) # Vain 100 havaintoa alkuperaisesta 507:sta
data <- bdims[data_i,]

# Eri menetelmien tarkkuuksien vertailuun ristiinvalidoinnilla tarvitaan foldit, k
    = 5
set.seed(NULL)
set.seed(seed_luku)
k_foldit <- createFolds(data$wgt, k = 5, list = FALSE)

# Lambdat, joilla sovitetaan regularisointimenetelmat

lambdat_kokeiluun <- 10^seq(2, -3, by = -0.1)

# Regressiokertointen vertailuun kaytetaan koko ylla valittua 100 havaintoa
    opetusaineistona.
# Vertaillaan nain saatuja malleja viela mielenkiinnosta taysin uuteen otokseen, n
    = 50.

train <- data
loppu_data <- bdims[-data_i,]
set.seed(NULL)
set.seed(seed_luku)
test <- loppu_data[sample(nrow(loppu_data), 50),]

# Sarake 23 = wgt, vastemuuttuja
y_train <- train[23]
y_test <- test[23]
x_train <- train[-23]
x_test <- test[-23]

# Normalisoidaan

z_parametrit <- preProcess(train)
train_z <- predict(z_parametrit, train)
test_z <- predict(z_parametrit, test)

x_train_z <- train_z[-23]
y_train_z <- train_z[23]

x_test_z <- test_z[-23]
y_test_z <- test_z[23]

y_train_vector <- as.vector(y_train_z$wgt)

########## Lineaarinen malli ##########

### Ennustevirheiden vertailuun ###
```

```
cv_rmse_lm <- c()
cv_sigmat_lm <- c()
for (i in 1:5) {
  # Data split
  opetus <- data[k_foldit != i,]
  testaus <- data[k_foldit == i,]
  z_par <- preProcess(opetus)
  opetus_z <- predict(z_par, opetus)
  testaus_z <- predict(z_par, testaus)

  # Malli ja ennusteet
  malli <- lm(wgt ~ ., data = opetus_z)
  cv_sigmat_lm[i] <- summary(malli)$sigma
  ennusteet <- predict(malli, newdata = testaus_z)
  cv_rmse_lm[i] <- RMSE(ennusteet, testaus_z$wgt) # sigmat talteen myohempaan
    kayttoon
}
# Tulos
cat("LM -- CV RMSE:", sqrt(mean(cv_rmse_lm^2)))

### Regressiokertoimien vertailuun ###

taysi_lm <- lm(wgt ~ .,
               data = train_z)
summary(taysi_lm)

# Residuaalien SD
sigma <- summary(taysi_lm)$sigma

# Ennusteet
lm_ennusteet <- predict(taysi_lm, newdata = test_z)
RMSE(lm_ennusteet, test_z$wgt)

########## Harjaregressio - glmnet ##########

### Ennustevirheiden vertailuun ###

nested_cv_rmse_glmnet_ridge <- c()
for (o in 1:5) {
  outer_opetus = data[k_foldit != o,]
  outer_testaus = data[k_foldit == o,]
  kaikkien_kokeiltavien_lambdojen_rmse <- c()
  for (l in 1:length(lambdat_kokeiluun)) {
    kokeiltava_lambda <- lambdat_kokeiluun[l]
    kokeiltavan_lambdan_rmse <- c()
    set.seed(NULL)
    set.seed(seed_luku)
    inner_foldit <- createFolds(outer_opetus$wgt, k = 5, list = FALSE)
    for (i in 1:5) {
      # Datan normalisointi
      inner_opetus <- outer_opetus[inner_foldit != i,]
      inner_testaus <- outer_opetus[inner_foldit == i,]
      inner_z_par <- preProcess(inner_opetus)
      inner_opetus_z <- predict(inner_z_par, inner_opetus)
      inner_testaus_z <- predict(inner_z_par, inner_testaus)
      inner_x_opetus_z <- as.matrix(inner_opetus_z[-23])
      inner_y_opetus_z <- as.matrix(inner_opetus_z[23])
      inner_x_testaus_z <- as.matrix(inner_testaus_z[-23])

      #Malli kokeiltavalla lambdalla
      inner_malli <- glmnet(x = inner_x_opetus_z, y = inner_y_opetus_z,
                            family = 'gaussian',
                            alpha = 0,
                            intercept = FALSE,
                            lambda = kokeiltava_lambda)

      # Ennusteet kokeiltavalla lambdalla
      inner_ennusteet <- predict(inner_malli, s = kokeiltava_lambda, newx = inner_x
    _testaus_z)
```

```
      kokeiltavan_lambdan_rmse[i] <- RMSE(inner_ennusteet, inner_testaus_z$wgt)
    }
    kaikkien_kokeiltavien_lambdojen_rmse[l] <- sqrt(mean(kokeiltavan_lambdan_rmse
    ^2))
  }
  # Paras lambda
  paras_lambda <- lambdat_kokeiluun[match(min(kaikkien_kokeiltavien_lambdojen_rmse)
    , kaikkien_kokeiltavien_lambdojen_rmse)]

  # Datan normalisointi
  outer_z_par <- preProcess(outer_opetus)
  outer_opetus_z <- predict(outer_z_par, outer_opetus)
  outer_testaus_z <- predict(outer_z_par, outer_testaus)
  outer_x_opetus_z <- as.matrix(outer_opetus_z[-23])
  outer_y_opetus_z <- as.matrix(outer_opetus_z[23])
  outer_x_testaus_z <- as.matrix(outer_testaus_z[-23])

  # Malli parhalla lambdalla
  outer_malli <- glmnet(x = outer_x_opetus_z, y = outer_y_opetus_z,
                        family = 'gaussian',
                        alpha = 0,
                        intercept = FALSE,
                        lambda = paras_lambda)

  # Ennusteet parhaalla lambdalla
  outer_ennusteet <- predict(outer_malli, s = paras_lambda, newx = outer_x_testaus_
    z)
  nested_cv_rmse_glmnet_ridge[o] <- RMSE(outer_ennusteet, outer_testaus_z$wgt)
}
# Tulos
cat("GLMNET-RIDGE -- NESTED-CV RMSE:", sqrt(mean(nested_cv_rmse_glmnet_ridge^2)) )

### Paras malli regressiokertoimien vertailuun ###

# Valitaan lambda
cv_rmse_glmnet_ridge <- c()
for (l in 1:length(lambdat_kokeiluun)) {
  kokeiltava_lambda = lambdat_kokeiluun[l]
  kokeiltavan_lambdan_rmse <- c()
  for (k in 1:5) {
    # Data split
    opetus <- data[k_foldit != k,]
    testaus <- data[k_foldit == k,]
    z_par <- preProcess(opetus)
    opetus_z <- predict(z_par, opetus)
    testaus_z <- predict(z_par, testaus)
    x_opetus_z <- as.matrix(opetus_z[-23])
    y_opetus_z <- as.matrix(opetus_z[23])
    x_testaus_z <- as.matrix(testaus_z[-23])

    # Malli kokeiltavalla lambdalla
    malli <- glmnet(x = x_opetus_z, y = y_opetus_z,
                    family = 'gaussian',
                    alpha = 0,
                    intercept = FALSE,
                    lambda = kokeiltava_lambda)

    # Ennusteet
    ennusteet <- predict(malli, s = kokeiltava_lambda, newx = x_testaus_z)
    kokeiltavan_lambdan_rmse[k] <- RMSE(ennusteet, testaus_z$wgt)
  }
  # Keskimaarainen RMSE kokeiltavalla lambdalla
  cv_rmse_glmnet_ridge[l] <- sqrt(mean(kokeiltavan_lambdan_rmse^2))
}
ridge_glmnet_lambda <- lambdat_kokeiluun[match(min(cv_rmse_glmnet_ridge), cv_rmse_
    glmnet_ridge)]
ridge_glmnet_rmse <- cv_rmse_glmnet_ridge[match(min(cv_rmse_glmnet_ridge), cv_rmse_
    glmnet_ridge)]

# Tulokset
```

```
cat("GLMNET-RIDGE -- Paras lambda:", ridge_glmnet_lambda, "-- RMSE:", ridge_glmnet_
    rmse)


# Mallin kertoimet ja RMSE parhaalla lambdalla
ridge_glmnet <- glmnet(x = as.matrix(x_train_z),
                       y = as.matrix(y_train_z),
                       family = 'gaussian',
                       alpha = 0,
                       intercept = FALSE,
                       lambda = ridge_glmnet_lambda)


# Kertoimet
coef(ridge_glmnet, s = ridge_glmnet_lambda)

# Ennusteet
ridge_glmnet_ennusteet <- predict(ridge_glmnet, s = ridge_glmnet_lambda, newx = as.
    matrix(x_test_z))
RMSE(ridge_glmnet_ennusteet, test_z$wgt)

########## Harjaregressio - MASS ##########

### Ennustevirheiden vertailuun ###

nested_cv_rmse_mass_ridge <- c()
for (o in 1:5) {
  outer_opetus = data[k_foldit != o,]
  outer_testaus = data[k_foldit == o,]
  kaikkien_kokeiltavien_lambdojen_rmse <- c()
  for (l in 1:length(lambdat_kokeiluun)) {
    kokeiltava_lambda <- lambdat_kokeiluun[l]
    kokeiltavan_lambdan_rmse <- c()
    set.seed(NULL)
    set.seed(seed_luku)
    inner_foldit <- createFolds(outer_opetus$wgt, k = 5, list = FALSE)
    for (i in 1:5) {
      # Datan normalisointi
      inner_opetus <- outer_opetus[inner_foldit != i,]
      inner_testaus <- outer_opetus[inner_foldit == i,]
      inner_z_par <- preProcess(inner_opetus)
      inner_opetus_z <- predict(inner_z_par, inner_opetus)
      inner_testaus_z <- predict(inner_z_par, inner_testaus)
      inner_x_testaus_z <- as.matrix(inner_testaus_z[-23])

      #Malli kokeiltavalla lambdalla
      inner_malli <- lm.ridge(wgt ~ .,
                              data = inner_opetus_z,
                              lambda = kokeiltava_lambda)

      # Ennusteet kokeiltavalla lambdalla
      inner_ennusteet <- inner_x_testaus_z %*% inner_malli$coef
      kokeiltavan_lambdan_rmse[i] <- RMSE(inner_ennusteet, inner_testaus_z$wgt)
    }
    kaikkien_kokeiltavien_lambdojen_rmse[l] <- sqrt(mean(kokeiltavan_lambdan_rmse
    ^2))
  }
  # Paras lambda
  paras_lambda <- lambdat_kokeiluun[match(min(kaikkien_kokeiltavien_lambdojen_rmse)
    , kaikkien_kokeiltavien_lambdojen_rmse)]

  # Datan normalisointi
  outer_z_par <- preProcess(outer_opetus)
  outer_opetus_z <- predict(outer_z_par, outer_opetus)
  outer_testaus_z <- predict(outer_z_par, outer_testaus)
  outer_x_testaus_z <- as.matrix(outer_testaus_z[-23])

  # Malli parhalla lambdalla
  outer_malli <- lm.ridge(wgt ~ .,
                          data = outer_opetus_z,
```

```
                          lambda = paras_lambda)

  # Ennusteet parhaalla lambdalla
  outer_ennusteet <- outer_x_testaus_z %*% outer_malli$coef
  nested_cv_rmse_mass_ridge[o] <- RMSE(outer_ennusteet, outer_testaus_z$wgt)
}
## Tulos
cat("MASS-RIDGE -- NESTED-CV RMSE:", sqrt(mean(nested_cv_rmse_mass_ridge^2)) )

### Paras malli regressiokertoimien vertailuun ###

# Valitaan lambda
cv_rmse_mass_ridge <- c()
for (l in 1:length(lambdat_kokeiluun)) {
  kokeiltava_lambda = lambdat_kokeiluun[l]
  kokeiltavan_lambdan_rmse <- c()
  for (k in 1:5) {
    # Data split
    opetus <- data[k_foldit != k,]
    testaus <- data[k_foldit == k,]
    z_par <- preProcess(opetus)
    opetus_z <- predict(z_par, opetus)
    testaus_z <- predict(z_par, testaus)
    x_testaus_z <- as.matrix(testaus_z[-23])

    # Malli kokeiltavalla lambdalla
    malli <- lm.ridge(wgt ~ .,
                      data = opetus_z,
                      lambda = kokeiltava_lambda)

    # Ennusteet
    ennusteet <- x_testaus_z %*% malli$coef
    kokeiltavan_lambdan_rmse[k] <- RMSE(ennusteet, testaus_z$wgt)
  }
  # Keskimaarainen RMSE kokeiltavalla lambdalla
  cv_rmse_mass_ridge[l] <- sqrt(mean(kokeiltavan_lambdan_rmse^2))
}
ridge_mass_lambda <- lambdat_kokeiluun[match(min(cv_rmse_mass_ridge), cv_rmse_mass_
    ridge)]
ridge_mass_rmse <- cv_rmse_mass_ridge[match(min(cv_rmse_mass_ridge), cv_rmse_mass_
    ridge)]

# Tulokset
cat("MASS-RIDGE -- Paras lambda:", ridge_mass_lambda, "-- RMSE:", ridge_mass_rmse)

# Mallin kertoimet ja RMSE parhaalla lambdalla
ridge_mass <- lm.ridge(wgt ~ ., data = train_z, lambda = ridge_mass_lambda)

# Kertoimet
ridge_mass$coef

# Ennusteet
ridge_mass_ennusteet <- as.matrix(x_test_z) %*% ridge_mass$coef
RMSE(ridge_mass_ennusteet, test_z$wgt)

########## Lasso - glmnet ##########

### Ennustevirheiden vertailuun ###

nested_cv_rmse_glmnet_lasso <- c()
for (o in 1:5) {
  outer_opetus = data[k_foldit != o,]
  outer_testaus = data[k_foldit == o,]
  kaikkien_kokeiltavien_lambdojen_rmse <- c()
  for (l in 1:length(lambdat_kokeiluun)) {
    kokeiltava_lambda <- lambdat_kokeiluun[l]
    kokeiltavan_lambdan_rmse <- c()
    set.seed(NULL)
    set.seed(seed_luku)
    inner_foldit <- createFolds(outer_opetus$wgt, k = 5, list = FALSE)
```

```
    for (i in 1:5) {
      # Datan normalisointi
      inner_opetus <- outer_opetus[inner_foldit != i,]
      inner_testaus <- outer_opetus[inner_foldit == i,]
      inner_z_par <- preProcess(inner_opetus)
      inner_opetus_z <- predict(inner_z_par, inner_opetus)
      inner_testaus_z <- predict(inner_z_par, inner_testaus)
      inner_x_opetus_z <- as.matrix(inner_opetus_z[-23])
      inner_y_opetus_z <- as.matrix(inner_opetus_z[23])
      inner_x_testaus_z <- as.matrix(inner_testaus_z[-23])

      #Malli kokeiltavalla lambdalla
      inner_malli <- glmnet(x = inner_x_opetus_z, y = inner_y_opetus_z,
                            family = 'gaussian',
                            alpha = 1,
                            intercept = FALSE,
                            lambda = kokeiltava_lambda)

      # Ennusteet kokeiltavalla lambdalla
      inner_ennusteet <- predict(inner_malli, s = kokeiltava_lambda, newx = inner_x
    _testaus_z)
      kokeiltavan_lambdan_rmse[i] <- RMSE(inner_ennusteet, inner_testaus_z$wgt)
    }
    kaikkien_kokeiltavien_lambdojen_rmse[l] <- sqrt(mean(kokeiltavan_lambdan_rmse
    ^2))
  }
  # Paras lambda
  paras_lambda <- lambdat_kokeiluun[match(min(kaikkien_kokeiltavien_lambdojen_rmse)
    , kaikkien_kokeiltavien_lambdojen_rmse)]

  # Datan normalisointi
  outer_z_par <- preProcess(outer_opetus)
  outer_opetus_z <- predict(outer_z_par, outer_opetus)
  outer_testaus_z <- predict(outer_z_par, outer_testaus)
  outer_x_opetus_z <- as.matrix(outer_opetus_z[-23])
  outer_y_opetus_z <- as.matrix(outer_opetus_z[23])
  outer_x_testaus_z <- as.matrix(outer_testaus_z[-23])

  # Malli parhalla lambdalla
  outer_malli <- glmnet(x = outer_x_opetus_z, y = outer_y_opetus_z,
                        family = 'gaussian',
                        alpha = 1,
                        intercept = FALSE,
                        lambda = paras_lambda)

  # Ennusteet parhaalla lambdalla
  outer_ennusteet <- predict(outer_malli, s = paras_lambda, newx = outer_x_testaus_
    z)
  nested_cv_rmse_glmnet_lasso[o] <- RMSE(outer_ennusteet, outer_testaus_z$wgt)
}
# Tulos
cat("GLMNET-LASSO -- NESTED-CV RMSE:", sqrt(mean(nested_cv_rmse_glmnet_lasso^2)))

### Paras malli regressiokertoimien vertailuun ###

cv_rmse_glmnet_lasso <- c()
for (l in 1:length(lambdat_kokeiluun)) {
  kokeiltava_lambda = lambdat_kokeiluun[l]
  kokeiltavan_lambdan_rmse <- c()
  for (k in 1:5) {
    # Data split
    opetus <- data[k_foldit != k,]
    testaus <- data[k_foldit == k,]
    z_par <- preProcess(opetus)
    opetus_z <- predict(z_par, opetus)
    testaus_z <- predict(z_par, testaus)
    x_opetus_z <- as.matrix(opetus_z[-23])
    y_opetus_z <- as.matrix(opetus_z[23])
    x_testaus_z <- as.matrix(testaus_z[-23])
```

```
    # Malli kokeiltavalla lambdalla
    malli <- glmnet(x = x_opetus_z, y = y_opetus_z,
                    family = 'gaussian',
                    alpha = 1,
                    intercept = FALSE,
                    lambda = kokeiltava_lambda)

    # Ennusteet
    ennusteet <- predict(malli, s = kokeiltava_lambda, newx = x_testaus_z)
    kokeiltavan_lambdan_rmse[k] <- RMSE(ennusteet, testaus_z$wgt)
  }
  # Keskimaarainen RMSE kokeiltavalla lambdalla
  cv_rmse_glmnet_lasso[l] <- sqrt(mean(kokeiltavan_lambdan_rmse^2))
}
lasso_glmnet_lambda <- lambdat_kokeiluun[match(min(cv_rmse_glmnet_lasso), cv_rmse_
    glmnet_lasso)]
lasso_glmnet_rmse <- cv_rmse_glmnet_lasso[match(min(cv_rmse_glmnet_lasso), cv_rmse_
    glmnet_lasso)]

# Tulokset
cat("GLMNET-lasso -- Paras lambda:", lasso_glmnet_lambda, "-- RMSE:", lasso_glmnet_
    rmse)

# Mallin kertoimet ja RMSE parhaalla lambdalla
lasso_glmnet <- glmnet(x = as.matrix(x_train_z),
                       y = as.matrix(y_train_z),
                       family = 'gaussian',
                       alpha = 1,
                       intercept = FALSE,
                       lambda = lasso_glmnet_lambda)

# Kertoimet
coef(lasso_glmnet, s = lasso_glmnet_lambda)

# Ennusteet
lasso_glmnet_ennusteet <- predict(lasso_glmnet, s = lasso_glmnet_lambda, newx = as.
    matrix(x_test_z))
RMSE(lasso_glmnet_ennusteet, test_z$wgt)

########## Lasso - lars ##########

### Ennustevirheiden vertailuun ###

nested_cv_rmse_lars_lasso <- c()
for (o in 1:5) {
  outer_opetus = data[k_foldit != o,]
  outer_testaus = data[k_foldit == o,]
  kaikkien_kokeiltavien_lambdojen_rmse <- c()
  for (l in 1:length(lambdat_kokeiluun)) {
    kokeiltava_lambda <- lambdat_kokeiluun[l]
    kokeiltavan_lambdan_rmse <- c()
    set.seed(NULL)
    set.seed(seed_luku)
    inner_foldit <- createFolds(outer_opetus$wgt, k = 5, list = FALSE)
    for (i in 1:5) {
      # Datan normalisointi
      inner_opetus <- outer_opetus[inner_foldit != i,]
      inner_testaus <- outer_opetus[inner_foldit == i,]
      inner_z_par <- preProcess(inner_opetus)
      inner_opetus_z <- predict(inner_z_par, inner_opetus)
      inner_testaus_z <- predict(inner_z_par, inner_testaus)
      inner_x_opetus_z <- as.matrix(inner_opetus_z[-23])
      inner_y_opetus_z <- as.matrix(inner_opetus_z[23])
      inner_x_testaus_z <- as.matrix(inner_testaus_z[-23])

      # Malli
      inner_malli <- lars(x = inner_x_opetus_z, y = inner_y_opetus_z,
                          type = "lasso",
                          normalize = FALSE,
                          intercept = FALSE)
```

```
      # Ennusteet kokeiltavalla lambdalla
      inner_ennusteet <- predict.lars(inner_malli, newx = inner_x_testaus_z, s =
    kokeiltava_lambda, type = "fit", mode = "lambda")
      kokeiltavan_lambdan_rmse[i] <- RMSE(inner_ennusteet$fit, inner_testaus_z$wgt)
    }
    kaikkien_kokeiltavien_lambdojen_rmse[l] <- sqrt(mean(kokeiltavan_lambdan_rmse
    ^2))
  }
  # Paras lambda
  paras_lambda <- lambdat_kokeiluun[match(min(kaikkien_kokeiltavien_lambdojen_rmse)
    , kaikkien_kokeiltavien_lambdojen_rmse)]

  # Datan normalisointi
  outer_z_par <- preProcess(outer_opetus)
  outer_opetus_z <- predict(outer_z_par, outer_opetus)
  outer_testaus_z <- predict(outer_z_par, outer_testaus)
  outer_x_opetus_z <- as.matrix(outer_opetus_z[-23])
  outer_y_opetus_z <- as.matrix(outer_opetus_z[23])
  outer_x_testaus_z <- as.matrix(outer_testaus_z[-23])

  # Malli
  outer_malli <- lars(x = outer_x_opetus_z, y = outer_y_opetus_z,
                      type = "lasso",
                      normalize = FALSE,
                      intercept = FALSE)

  # Ennusteet parhaalla lambdalla
  outer_ennusteet <- predict.lars(outer_malli, newx = outer_x_testaus_z, s = paras_
    lambda, type = "fit", mode = "lambda")
  nested_cv_rmse_lars_lasso[o] <- RMSE(outer_ennusteet$fit, outer_testaus_z$wgt)
}
# Tulos
cat("LARS-LASSO -- NESTED-CV RMSE:", sqrt(mean(nested_cv_rmse_lars_lasso^2)))

### Paras malli regressiokertoimien vertailuun ###

cv_rmse_lars_lasso <- c()
for (l in 1:length(lambdat_kokeiluun)) {
  kokeiltava_lambda = lambdat_kokeiluun[l]
  kokeiltavan_lambdan_rmse <- c()
  for (k in 1:5) {
    # Data split
    opetus <- data[k_foldit != k,]
    testaus <- data[k_foldit == k,]
    z_par <- preProcess(opetus)
    opetus_z <- predict(z_par, opetus)
    testaus_z <- predict(z_par, testaus)
    x_opetus_z <- as.matrix(opetus_z[-23])
    y_opetus_z <- as.matrix(opetus_z[23])
    x_testaus_z <- as.matrix(testaus_z[-23])

    # Malli
    malli <- lars(x = x_opetus_z, y = y_opetus_z,
                  type = "lasso",
                  normalize = FALSE,
                  intercept = FALSE)

    # Ennusteet kokeiltavalla lambdalla
    ennusteet <- predict.lars(malli, newx = x_testaus_z, s = kokeiltava_lambda,
    type = "fit", mode = "lambda")
    kokeiltavan_lambdan_rmse[k] <- RMSE(ennusteet$fit, testaus_z$wgt)
  }
  # Keskimaarainen RMSE kokeiltavalla lambdalla
  cv_rmse_lars_lasso[l] <- sqrt(mean(kokeiltavan_lambdan_rmse^2))
}
lasso_lars_lambda <- lambdat_kokeiluun[match(min(cv_rmse_lars_lasso), cv_rmse_lars_
    lasso)]
lasso_lars_rmse <- cv_rmse_lars_lasso[match(min(cv_rmse_lars_lasso), cv_rmse_lars_
    lasso)]
```

```
# Tulokset
cat("LARS-lasso -- Paras lambda:", lasso_lars_lambda, "-- RMSE:", lasso_lars_rmse)

# Mallin kertoimet ja RMSE parhaalla lambdalla
lasso_lars <- lars(x = as.matrix(x_train_z),
                   y = as.matrix(y_train_z),
                   type = "lasso",
                   normalize = FALSE,
                   intercept = FALSE)

# Kertoimet parhaalla lambdalla
predict.lars(lasso_lars, type = "coefficients", mode = "lambda", s = lasso_lars_
    lambda)$coefficients

# Ennusteet
lasso_lars_ennusteet <- predict.lars(lasso_lars, newx = x_test_z, s = lasso_lars_
    lambda, type = "fit", mode = "lambda")
RMSE(lasso_lars_ennusteet$fit, test_z$wgt)

########## MAP - Harjaregressio: glmnet-lambda ##########

# Ridgemalli STAN-tiedostosta
ridge_stan <- stan_model(file = 'ridge_stan.stan')

### Regressiokertoimien vertailuun ###

set.seed(NULL)
set.seed(seed_luku)
ridge_map_glmnet <- optimizing(ridge_stan,
                               data = list(x_train = as.matrix(x_train_z),
                                           n = nrow(as.matrix(x_train_z)),
                                           p = ncol(as.matrix(x_train_z)),
                                           y_train = y_train_vector,
                                           lambda = nrow(as.matrix(x_train_z)) *
    ridge_glmnet_lambda,
                                           sigma_e = sigma)
)

# Kertoimet
ridge_map_glmnet$par

# Ennusteet
ridge_map_glmnet_ennusteet <- as.matrix(x_test_z) %*% ridge_map_glmnet$par
RMSE(ridge_map_glmnet_ennusteet, test_z$wgt)

########## MAP - Harjaregressio: MASS-lambda ##########

### Regressiokertoimien vertailuun ###

set.seed(NULL)
set.seed(seed_luku)
ridge_map_mass <- optimizing(ridge_stan,
                             data = list(x_train = as.matrix(x_train_z),
                                         n = nrow(as.matrix(x_train_z)),
                                         p = ncol(as.matrix(x_train_z)),
                                         y_train = y_train_vector,
                                         lambda = ridge_mass_lambda,
                                         sigma_e = sigma)
)

# Kertoimet
ridge_map_mass$par

# Ennusteet
ridge_map_mass_ennusteet <- as.matrix(x_test_z) %*% ridge_map_mass$par
RMSE(ridge_map_mass_ennusteet, test_z$wgt)

########## MAP - Harjaregressio: oma lambda ##########
```

```
### Paras malli regressiokertoimien vertailuun ###

cv_rmse_map_ridge <- c()
for (l in 1:length(lambdat_kokeiluun)) {
  kokeiltava_lambda = lambdat_kokeiluun[l]
  kokeiltavan_lambdan_rmse <- c()
  for (i in 1:5) {
    # Data split
    opetus <- data[k_foldit != i,]
    testaus <- data[k_foldit == i,]
    z_par <- preProcess(opetus)
    opetus_z <- predict(z_par, opetus)
    testaus_z <- predict(z_par, testaus)
    x_opetus_z <- as.matrix(opetus_z[-23])
    y_opetus_z <- as.vector(opetus_z$wgt)
    x_testaus_z <- as.matrix(testaus_z[-23])

    # Malli
    set.seed(NULL)
    set.seed(seed_luku)
    optimointi <- optimizing(ridge_stan,
                             data = list(x_train = x_opetus_z,
                                         n = nrow(x_opetus_z),
                                         p = ncol(x_opetus_z),
                                         y_train = y_opetus_z,
                                         lambda = kokeiltava_lambda,
                                         sigma_e = cv_sigmat_lm[i])
    )

    # Ennusteet
    ennusteet <- x_testaus_z %*% optimointi$par
    kokeiltavan_lambdan_rmse[i] <- RMSE(ennusteet, testaus_z$wgt)
  }
  # Keskimaarainen RMSE kokeiltavalla lambdalla
  cv_rmse_map_ridge[l] <- sqrt(mean(kokeiltavan_lambdan_rmse^2))
}
ridge_map_lambda <- lambdat_kokeiluun[match(min(cv_rmse_map_ridge), cv_rmse_map_
    ridge)]
ridge_map_rmse <- cv_rmse_map_ridge[match(min(cv_rmse_map_ridge), cv_rmse_map_ridge
    )]

# Tulokset
cat("MAP-RIDGE -- Lambda", ridge_map_lambda, "-- RMSE:", ridge_map_rmse)

# Mallin kertoimet ja RMSE parhaalla lambdalla
set.seed(NULL)
set.seed(seed_luku)
ridge_map <- optimizing(ridge_stan,
                        data = list(x_train = as.matrix(x_train_z),
                                    n = nrow(as.matrix(x_train_z)),
                                    p = ncol(as.matrix(x_train_z)),
                                    y_train = y_train_vector,
                                    lambda = ridge_map_lambda,
                                    sigma_e = sigma)
)

# Kertoimet
ridge_map$par

# Ennusteet
ridge_map_ennusteet <- as.matrix(x_test_z) %*% ridge_map$par
RMSE(ridge_map_ennusteet, test_z$wgt)

########## MAP - Lasso: glmnet-lambda ##########

# Lassomalli STAN-tiedostosta
lasso_stan <- stan_model(file = 'lasso_stan.stan')

### Regressiokertoimien vertailuun ###
```

```
set.seed(NULL)
set.seed(seed_luku)
lasso_map_glmnet <- optimizing(lasso_stan,
                               data = list(x_train = as.matrix(x_train_z),
                                           n = nrow(as.matrix(x_train_z)),
                                           p = ncol(as.matrix(x_train_z)),
                                           y_train = y_train_vector,
                                           lambda = nrow(as.matrix(x_train_z)) *
    lasso_glmnet_lambda,
                                           sigma_e = sigma)
)

# Kertoimet
lasso_map_glmnet$par

# Ennusteet
lasso_map_glmnet_ennusteet <- as.matrix(x_test_z) %*% lasso_map_glmnet$par
RMSE(lasso_map_glmnet_ennusteet, test_z$wgt)

########## MAP - Lasso: lars-lambda ##########

### Regressiokertoimien vertailuun ###

set.seed(NULL)
set.seed(seed_luku)
lasso_map_lars <- optimizing(lasso_stan,
                             data = list(x_train = as.matrix(x_train_z),
                                         n = nrow(as.matrix(x_train_z)),
                                         p = ncol(as.matrix(x_train_z)),
                                         y_train = y_train_vector,
                                         lambda = lasso_lars_lambda,
                                         sigma_e = sigma)
)

# Kertoimet
lasso_map_lars$par

# Ennusteet
lasso_map_lars_ennusteet <- as.matrix(x_test_z) %*% lasso_map_lars$par
RMSE(lasso_map_lars_ennusteet, test_z$wgt)

########## MAP - Lasso: oma lambda ##########

### Paras malli regressiokertoimien vertailuun ###

cv_rmse_map_lasso <- c()
for (l in 1:length(lambdat_kokeiluun)) {
  kokeiltava_lambda = lambdat_kokeiluun[l]
  kokeiltavan_lambdan_rmse <- c()
  for (i in 1:5) {
    # Data split
    opetus <- data[k_foldit != i,]
    testaus <- data[k_foldit == i,]
    z_par <- preProcess(opetus)
    opetus_z <- predict(z_par, opetus)
    testaus_z <- predict(z_par, testaus)
    x_opetus_z <- as.matrix(opetus_z[-23])
    y_opetus_z <- as.vector(opetus_z$wgt)
    x_testaus_z <- as.matrix(testaus_z[-23])

    # Malli
    set.seed(NULL)
    set.seed(seed_luku)
    optimointi <- optimizing(lasso_stan,
                             data = list(x_train = x_opetus_z,
                                         n = nrow(x_opetus_z),
                                         p = ncol(x_opetus_z),
                                         y_train = y_opetus_z,
                                         lambda = kokeiltava_lambda,
                                         sigma_e = cv_sigmat_lm[i])
```

```
    )

    # Ennusteet
    ennusteet <- x_testaus_z %*% optimointi$par
    kokeiltavan_lambdan_rmse[i] <- RMSE(ennusteet, testaus_z$wgt)
  }
  # Keskimaarainen RMSE kokeiltavalla lambdalla
  cv_rmse_map_lasso[l] <- sqrt(mean(kokeiltavan_lambdan_rmse^2))
}
lasso_map_lambda <- lambdat_kokeiluun[match(min(cv_rmse_map_lasso), cv_rmse_map_
    lasso)]
lasso_map_rmse <- cv_rmse_map_lasso[match(min(cv_rmse_map_lasso), cv_rmse_map_lasso
    )]

# Tulokset
cat("MAP-LASSO -- Lambda", lasso_map_lambda, "-- RMSE:", lasso_map_rmse)

# Mallin kertoimet ja RMSE parhaalla lambdalla
set.seed(NULL)
set.seed(seed_luku)
lasso_map <- optimizing(lasso_stan,
                        data = list(x_train = as.matrix(x_train_z),
                                    n = nrow(as.matrix(x_train_z)),
                                    p = ncol(as.matrix(x_train_z)),
                                    y_train = y_train_vector,
                                    lambda = lasso_map_lambda,
                                    sigma_e = sigma)
)

# Kertoimet
lasso_map$par

# Ennusteet
lasso_map_ennusteet <- as.matrix(x_test_z) %*% lasso_map$par
RMSE(lasso_map_ennusteet, test_z$wgt)

########## Otosmediaanit - harjaregressio: glmnet-lambda ##########

### Regressiokertoimien vertailuun ###

set.seed(NULL)
set.seed(seed_luku)
ridge_sample_glmnet <- sampling(ridge_stan,
                                data = list(x_train = as.matrix(x_train_z),
                                            n = nrow(as.matrix(x_train_z)),
                                            p = ncol(as.matrix(x_train_z)),
                                            y_train = y_train_vector,
                                            lambda = ridge_glmnet_lambda * nrow(x_
    train_z),
                                            sigma_e = sigma))

# Otos ja kertoimet
ridge_sample_glmnet_otos <- rstan::extract(ridge_sample_glmnet)
colMedians(ridge_sample_glmnet_otos$beta)

# Ennusteet
ridge_sample_glmnet_ennusteet <- as.matrix(x_test_z) %*% colMedians(ridge_sample_
    glmnet_otos$beta)
RMSE(ridge_sample_glmnet_ennusteet, test_z$wgt)

########## Otosmediaanit - harjaregressio: mass-lambda ##########

### Regressiokertoimien vertailuun ###

set.seed(NULL)
set.seed(seed_luku)
ridge_sample_mass <- sampling(ridge_stan,
                              data = list(x_train = as.matrix(x_train_z),
                                          n = nrow(as.matrix(x_train_z)),
                                          p = ncol(as.matrix(x_train_z)),
```

```
                                        y_train = y_train_vector,
                                        lambda = ridge_mass_lambda,
                                        sigma_e = sigma))

# Otos ja kertoimet
ridge_sample_mass_otos <- rstan::extract(ridge_sample_mass)
colMedians(ridge_sample_mass_otos$beta)

# Ennusteet
ridge_sample_mass_ennusteet <- as.matrix(x_test_z) %*% colMedians(ridge_sample_mass
    _otos$beta)
RMSE(ridge_sample_mass_ennusteet, test_z$wgt)

########## Otosmediaanit - lasso: glmnet-lambda ##########

### Regressiokertoimien vertailuun ###

set.seed(NULL)
set.seed(seed_luku)
lasso_sample_glmnet <- sampling(lasso_stan,
                                data = list(x_train = as.matrix(x_train_z),
                                            n = nrow(as.matrix(x_train_z)),
                                            p = ncol(as.matrix(x_train_z)),
                                            y_train = y_train_vector,
                                            lambda = nrow(as.matrix(x_train_z)) *
    lasso_glmnet_lambda,
                                            sigma_e = sigma))

# Otos ja kertoimet
lasso_sample_glmnet_otos <- rstan::extract(lasso_sample_glmnet)
colMedians(lasso_sample_glmnet_otos$beta)

# Ennusteet
lasso_sample_glmnet_ennusteet <- as.matrix(x_test_z) %*% colMedians(lasso_sample_
    glmnet_otos$beta)
RMSE(lasso_sample_glmnet_ennusteet, test_z$wgt)

########## Otosmediaanit - lasso: lars-lambda ##########

### Regressiokertoimien vertailuun ###

set.seed(NULL)
set.seed(seed_luku)
lasso_sample_lars <- sampling(lasso_stan,
                              data = list(x_train = as.matrix(x_train_z),
                                          n = nrow(as.matrix(x_train_z)),
                                          p = ncol(as.matrix(x_train_z)),
                                          y_train = y_train_vector,
                                          lambda = lasso_lars_lambda,
                                          sigma_e = sigma))

# Otos ja kertoimet
lasso_sample_lars_otos <- rstan::extract(lasso_sample_lars)
colMedians(lasso_sample_lars_otos$beta)

# Ennusteet
lasso_sample_lars_ennusteet <- as.matrix(x_test_z) %*% colMedians(lasso_sample_lars
    _otos$beta)
RMSE(lasso_sample_lars_ennusteet, test_z$wgt)

########## Regressiokertoimien vertailu kaikilla lambdoilla ##########

# Harjaregressio

ridge_virhe_map_vs_glmnet <- c()
ridge_virhe_map_vs_mass <- c()
ridge_virhe_glmnet_vs_mass <- c()
ridge_virhe_kasin_vs_glmnet <- c()
ridge_virhe_kasin_vs_mass <- c()
ridge_virhe_kasin_vs_map <- c()
```

```
for (l in 1:length(lambdat_kokeiluun)) {
  kokeiltava_lambda <- lambdat_kokeiluun[l]

  # Malli1
  set.seed(NULL)
  set.seed(seed_luku)
  map <- optimizing(ridge_stan,
                    data = list(x_train = x_train_z,
                                n = nrow(x_train_z),
                                p = ncol(x_train_z),
                                y_train = y_train_vector,
                                lambda = kokeiltava_lambda,
                                sigma_e = sigma)
  )
  # Malli2
  frek1 <- glmnet(x = as.matrix(x_train_z),
                  y = as.matrix(y_train_z),
                  family = 'gaussian',
                  alpha = 0,
                  intercept = FALSE,
                  lambda = kokeiltava_lambda / nrow(x_train_z))

  # Malli3 <-
  frek2 <- lm.ridge(wgt ~ .,
                    data = train_z,
                    lambda = kokeiltava_lambda)

  # Kasin ratkaistu ridge
  kasin <- solve(t(as.matrix(x_train_z)) %*% as.matrix(x_train_z) + kokeiltava_
    lambda * diag(ncol(as.matrix(x_train_z)))) %*% t(as.matrix(x_train_z)) %*% as.
    matrix(y_train_z)

  # Virhe
  ridge_virhe_map_vs_glmnet[l] <- RMSE(as.matrix(frek1$beta), as.matrix(map$par) )
  ridge_virhe_map_vs_mass[l] <- RMSE( as.matrix(frek2$coef), as.matrix(map$par) )
  ridge_virhe_glmnet_vs_mass[l] <- RMSE( as.matrix(frek2$coef), as.matrix(frek1$
    beta) )

  ridge_virhe_kasin_vs_glmnet[l] <- RMSE(as.matrix(kasin), as.matrix(frek1$beta))
  ridge_virhe_kasin_vs_mass[l] <- RMSE(as.matrix(kasin), as.matrix(frek2$coef))
  ridge_virhe_kasin_vs_map[l] <- RMSE(as.matrix(kasin), as.matrix(map$par))

}
sqrt(mean(ridge_virhe_map_vs_glmnet^2))
sqrt(mean(ridge_virhe_map_vs_mass^2))
sqrt(mean(ridge_virhe_glmnet_vs_mass^2))
sqrt(mean(ridge_virhe_kasin_vs_glmnet^2))
sqrt(mean(ridge_virhe_kasin_vs_mass^2))
sqrt(mean(ridge_virhe_kasin_vs_map^2))

# Lasso

lasso_virhe_map_vs_glmnet <- c()
lasso_virhe_map_vs_lars <- c()
lasso_virhe_glmnet_vs_lars <- c()
for (l in 1:length(lambdat_kokeiluun)) {

  kokeiltava_lambda <- lambdat_kokeiluun[l]

  # Malli1
  set.seed(NULL)
  set.seed(seed_luku)
  map <- optimizing(lasso_stan,
                    data = list(x_train = x_train_z,
                                n = nrow(x_train_z),
                                p = ncol(x_train_z),
                                y_train = y_train_vector,
                                lambda = kokeiltava_lambda,
                                sigma_e = sigma)
  )
```

```
  # Malli2
  frek1 <- glmnet(x = as.matrix(x_train_z),
                  y = as.matrix(y_train_z),
                  family = 'gaussian',
                  alpha = 0,
                  intercept = FALSE,
                  lambda = kokeiltava_lambda / nrow(x_train_z))

  # Malli3 <-
  frek2 <- lars(x = as.matrix(x_train_z),
                y = as.matrix(y_train_z),
                type = "lasso",
                normalize = FALSE,
                intercept = FALSE)

  # Virhe
  lasso_virhe_map_vs_glmnet[l] <- RMSE(as.matrix(frek1$beta), as.matrix(map$par) )
  lasso_virhe_map_vs_lars[l] <- RMSE( as.matrix(predict.lars(frek2, type = "
    coefficients", mode = "lambda", s = kokeiltava_lambda)$coefficients), as.matrix
    (map$par) )
  lasso_virhe_glmnet_vs_lars[l] <- RMSE( as.matrix(predict.lars(frek2, type = "
    coefficients", mode = "lambda", s = kokeiltava_lambda)$coefficients), as.matrix
    (frek1$beta) )

}
sqrt(mean(lasso_virhe_map_vs_glmnet^2))
sqrt(mean(lasso_virhe_map_vs_lars^2))
sqrt(mean(lasso_virhe_glmnet_vs_lars^2))
```

Liite B Stan-koodi

```
// Harjaregressio: ridge_stan.stan
data {
  int<lower=0> n;
  int<lower=0> p;
  matrix[n, p] x_train;
  vector[n] y_train;
  real<lower=0> lambda;
  real<lower=0> sigma_e;
}

parameters {
  vector[p] beta;
}

model {
  beta ~ normal(0,  sqrt( sigma_e^2 / lambda ) );
  y_train ~ normal(x_train * beta, sigma_e);
}

// Lasso: lasso_stan.stan
data {
  int<lower=0> n;
  int<lower=0> p;
  matrix[n, p] x_train;
  vector[n] y_train;
  real<lower=0> lambda;
  real<lower=0> sigma_e;
}

parameters {
  vector[p] beta;
}

model {
  beta ~ double_exponential(0, ( (2 * (sigma_e)^2 ) / lambda ) );
  y_train ~ normal(x_train * beta, sigma_e);
}
```