Tilastotieteen pro gradu -tutkielma

Luokittelumenetelmän evaluointimenetelmien vertailu pienten aineistojen tapauksessa simulointikokein

Lasse Moisio

Jyväskylän yliopisto
Matematiikan ja tilastotieteen laitos
24. marraskuuta 2016

Tiivistelmä

Lasse Moisio: *Luokittelumenetelmän evaluointimenetelmien vertailu pienten aineistojen tapauksessa simulointikokein*
Tilastotieteen pro gradu -tutkielma, 32 sivua + 3 liitettä (13 sivua). Jyväskylän yliopisto, Matematiikan ja tilastotieteen laitos, 24. marraskuuta 2016.

Luokitteluanalyysin tyypillinen ongelma on mallin valinta ja/tai muuttujien eli piirteiden valinta. Tavoitteena on estimoida luotettavasti luokitteluvirheen todennäköisyys opetusaineiston avulla – ideaalissa tilanteessa käyttäen riippumatonta testiaineistoa. Tällöin saadaan estimaatti yleistämisvirheelle, joka kuvaa opetusaineistosta estimoidun mallin kykyä ennustaa riippumatonta testiaineistoa. Usein kuitenkin aineiston niukkuudesta johtuen ei voida käyttää erillistä riippumatonta testiaineistoa, jolloin ei myöskään voida estimoida suoraan yleistämisvirhettä.

Tällöin voidaan kuitenkin käyttää otoksen uudelleenkäyttömenetelmiä, joista käytetyimmät lienevät ristiinvalidointi ja bootstrap. Menetelmien etuna on, että ne estimoivat hyvin odotettua yleistämisvirhettä, joka ei rajoitu vain tiettyyn opetusaineistoon, vaan on odotusarvo yli kaikkien mahdollisten opetusaineistojen. Odotettu yleistämisvirhe on osoittautunutkin useiden tutkimusten mukaan hyödylliseksi työkaluksi tilastollisessa analyysissa.

Luokitteluvirheen todennäköisyyden arviointiin liittyy olennaisesti myös käsite Bayes-virhe, jolla tarkoitetaan pienintä saavutettavissa olevaa luokitteluvirheen todennäköisyyttä. Tässä työssä piirteitä mallinnetaan p-ulotteisella normaalijakaumalla ja rajaudutaan kvadraattiseen luokittelijaan (QDA). Tyypillisesti Bayes-virhettä ei tiedetä, ja sen laskeminen analyyttisesti on usein haastavaa tai mahdotonta. Tässä työssä esitetään menetelmä Bayes-virheen arvioimiseksi Monte Carlo -integroinnilla.

Aiemmissa tutkimuksissa otoksen uudelleenkäyttömenetelmiä on verrattu odotettuun yleistämisvirheeseen, kuten myös tehdään tässä työssä. Lisäksi otoksen uudelleenkäyttömenetelmiä verrataan Bayes-virheeseen. Parhaat tulokset saavutettiin .632-bootstrap-estimaattorilla ja .632+-bootstrap-estimaattorilla sekä toistetulla ristiinvalidoinnilla K:n arvolla 10. Toistettu ristiinvalidointi K:n arvoilla 5 ja 10 ja aineiston jako toistuvasti opetus- ja testiaineistoksi jakosuhteilla 80/20 ja 90/10 tuottivat likimain yhtä hyviä tuloksia.

Menetelmien vertailu toteutettiin simulointikokeella, joka perustuu 50 luokkaa ja 64 piirrettä sisältävään pohjaeläinaineistoon. Simulointikokeessa käytettävät luokkien (2—8) ja piirteiden (2—50) lukumäärät ovat suurempia kuin aiemmissa tutkimuksissa on yleensä käytetty.

Avainsanat: Bayes-virhe, yleistämisvirhe, luokitteluvirhe, QDA, bootstrap, ristiinvalidointi, aineiston jako

Sisältö

1 Johdanto

Luokitteluanalyysin tyypillisenä lähtökohtana on jaotella havainnot niistä mitattujen muuttujien eli piirteiden avulla ennalta määrättyihin luokkiin. Tällöin puhutaan ohjatusta oppimisesta (*supervised learning*). Ohjaamattoman oppimisen (*unsupervised learning*) tapauksessa havaintojen luokat eivät ole tiedossa, jolloin havainnot pyritään luokittelemaan esimerkiksi klusteroinnin avulla. Tässä työssä keskitytään ainoastaan ohjattuun oppimiseen. Luokittelusäännön muodostamiseksi tarvitaan opetusaineisto, josta on mitattu havaintojen piirteet. Tällöin havaintojen todelliset luokat on täytynyt määrittää ulkoisen mekanismin avulla – esimerkiksi alan asiantuntijoiden avulla. Hahmontunnistusjärjestelmä voidaan ajatella monivaiheisena prosessina, johon kuuluvat aineiston keruu, esikäsittely, piirteiden irrottaminen ja luokittelu (Duda et al., 2000). Tässä työssä kiinnostuksen kohteena on prosessin viimeinen vaihe eli luokittelu ja sen hyvyyden arviointi.

Luokittelun hyvyyttä voidaan arvioida käyttämällä testiaineistoa, jonka havaintojen todelliset luokat ovat tiedossa. Luokittelu suoritetaan testiaineistosta mitattujen piirteiden avulla, jolloin päästään arvioimaan luokitteluvirheen todennäköisyyttä väärinluokiteltujen havaintojen osuudella. Luokittelumenetelmän hyvyydelle voidaan käyttää muitakin kriteereitä, kuten esimerkiksi luokitteluriskiä, jolloin luokittelusääntö voidaan muokata suosimaan joitakin tärkeitä luokkia. Tavoitteena on parhaan mallin ja/tai muuttujien eli piirteiden valinta, jolloin malli toimisi luotettavasti uusille samasta populaatiosta oleville riippumattomille otoksille. Esimerkiksi testataan empiirisesti käytössä olevalla aineistolla, mikä malli minimoi luokitteluvirheen todennäköisyyden, ja lopuksi estimoidaan yleistämisvirhe valitulla ”parhaalla” mallilla käyttäen riippumatonta testiaineistoa. Käytännön tilanteissa otos on usein pieni, eikä täten ole mahdollista käyttää riippumatonta testiaineistoa. Tällöin voidaan käyttää otoksen uudelleenkäyttömenetelmiä kuten ristiinvalidointia tai bootstrap-menetelmiä, jotka estimoivat suoraan odotettua yleistämisvirhettä.

Tässä tutkielmassa lähtökohtana on bayesiläinen päätöksentekoteoria, joka on eräs tilastollinen lähestymistapa hahmontunnistukseen. Lähestymistavalle ominaista on, että luokittelu suoritetaan perustuen todennäköisyyksiin ja luokittelupäätöksistä aiheutuviin tappioihin. Bayesin päätössääntö takaa luokittelijalle pienimmän keskimääräisen luokitteluvirheen todennäköisyyden. Luokitteluvirheen todennäköisyyden arviointiin liittyy olennaisesti käsite Bayes-virhe (Duda et al., 2000), jolla tarkoitetaan pienintä saavutettavissa olevaa luokitteluvirheen todennäköisyyttä annetulle piirteiden jakaumalle. Piirteiden jakaumien ollessa tiedossa Bayes-virheen laskeminen analyyttisesti on hankalaa etenkin korkeissa dimensioissa (Duda et al., 2000). Sen tähden tässä tutkielmassa Bayes-virhettä arvioidaan Monte Carlo -integroinnilla (Robert, C. & Casella, G., 2009). Yksinkertaisuuden vuoksi piirteitä mallinnetaan usein p-ulotteisella normaalijakaumalla, kuten tässäkin työssä. Tällöin päädytään lineaariseen luokittelijaan (LDA) ja kvadraattiseen luokittelijaan (QDA), joista QDA on erityisen mielenkiinnon kohteena.

Ristiinvalidointi on yksi tunnetuimmista otoksen uudelleenkäyttömenetelmistä. Sitä käytetään tilastotieteessä tyypillisesti ennustevirheen estimoimiseen aineistosta. Menetelmän perusideana on jakaa aineisto K:hon yhtä suureen osaan. Ensin sovitetaan malli $K-1$ osalle, jonka jälkeen lasketaan ennustevirhe pois jätetylle osalle.

Toimenpide suoritetaan K kertaa siten, että kullekin osalle vuorollaan estimoidaan ennustevirhe, ja lopuksi yhdistetään ennustevirheiden estimaatit. Ristiinvalidointi on luonnollinen lähestymistapa erityisesti luokittelun yhteydessä. Tässä tutkielmassa on tarkoituksena mm. vertailla ristiinvalidointia eri K:n arvoilla. Ristiinvalidoinnista on esitetty myös versio, jossa datan ositus toistetaan useita kertoja, jolloin estimaattorin vaihtelua saadaan pienennettyä (Kim, 2009).

Toinen lähestymistapa on bootstrap-menetelmä, jota voidaan käyttää esimerkiksi halutun tunnusluvun jakauman estimoimiseen aineistosta – erityisesti, kun se on analyyttisesti hankalaa. Bootstrap-menetelmän ideana on muodostaa otannalla bootstrap-otoksia takaisinsijoittaen alkuperäisestä aineistosta ja määrätä kullekin bootstrap-otokselle haluttu tunnusluku. Luokittelun yhteydessä bootstrap-menetelmän tarkoituksena on pienentää estimaattorin vaihtelua, mikä voisi toimia hyvin etenkin pienten aineistojen tapauksessa. Bootstrap-menetelmiä on tutkittu paljon. Vertailuissa hyvin ovat pärjänneet 0.632-bootstrap-estimaattori sekä korjattu 0.632+-bootstrap-estimaattori (Efron & Tibshirani, 1997), joita tutkitaan myös tässä työssä.

Kolmas lähestymistapa on aineiston jako toistuvasti opetus- ja testiaineistoksi, joka on myös paljon käytetty menetelmä. Opetus- ja testiaineistojakoa sekä ristiinvalidointia on vertailtu aiemmin simulointikokeilla (mm. töissä Molinaro et al., 2005 & Kim, 2009). Kim (2009) pitää toistettua ristiinvalidointia stabiilimpana, ja siten yleisesti suositellumpana menetelmänä. Molinaro et al. (2005) mukaan ei ole erityistä syytä suosia opetus- ja testiaineistojakoa.

Otoksen uudelleenkäyttömenetelmiä on vertailtu aiemmin simulointikokeilla (mm. töissä Molinaro et al., 2005; Kim, 2009; Borra & Di Ciaccio, 2010). Erona moniin aiempiin vastaaviin simulointikokeisiin tässä tutkielmassa simulointikoe perustuu todelliseen aineistoon, jolloin myös luokkien ja piirteiden määrä asetetaan vastaamaan todellista tilannetta, ts. luokkien määrä vaihtelee välillä 2—50 ja piirteiden määrä välillä 2—8. Myös käytettävät luokittelumenetelmät eroavat. Edellä mainituissa töissä käytettiin mm. luokittelumenetelmiä LDA, lähimmän naapurin menetelmä, regressiopuu ja neuroverkot. Aiemmissa tutkimuksissa otoksen uudelleenkäyttömenetelmiä on verrattu odotettuun yleistämisvirheen. Samoin tehdään tässä työssä, mutta otoksen uudelleenkäyttömenetelmiä verrataan myös Bayes-virheeseen.

Luvun 4 simulointikoe perustuu pohjaeläinaineistoon, joka esitellään samassa luvussa. Ympäristön seurannassa käytetään vesistöjen pohjissa eläviä pohjaeläimiä, jotka reagoivat herkästi ekosysteemin muutoksiin (Vuori et al., 2009). Pohjaeläimien tunnistaminen manuaalisesti on aikaavievää ja kallista, joten havaintojen koneellinen luokittelu antaa uusia mahdollisuuksia ympäristön tilan seurantaan.

Luvussa 2 esitetään bayesiläinen päätöksentekoteoria päätyen lopulta kvadraattiseen luokittelusääntöön (QDA). Luvussa 3 esitetään ensin analyyttinen tapa arvioida Bayes-virhettä kahden luokan tapauksessa. Tämän jälkeen Bayes-virhettä arvioidaan Monte Carlo -integroinnilla, ja lopuksi Bayes-virhettä arvioidaan erilaisilla otoksen uudelleenkäyttömenetelmillä. Luvussa 4 esitellään pohjaeläinaineisto ja siihen perustuva simulointikoe. Luvussa 5 esitetään yhteenveto saaduista tuloksista.

2 Bayesiläinen päätöksentekoteoria

Tämä luku perustuu pääasiassa kirjaan Duda et al. (2000). Bayesiläinen päätöksentekoteoria on eräs tilastollinen lähestymistapa hahmontunnistukseen. Ajatuksena on tehdä luokittelupäätös perustuen todennäköisyyksiin sekä päätöksistä aiheutuviin tappioihin. Oletetaan tilanne, jossa on K ennalta tunnettua luokkaa $\omega_1, \ldots, \omega_K$, joita voidaan kutsua myös luonnontiloiksi. Luonnontila $\omega = \omega_k$, $k = 1, \ldots, K$, ajatellaan muuttujaksi, jota kuvaillaan todennäköisyystermein.

Bayesiläinen päätöksentekoteoria on yksi Bayes-menetelmän sovelluksista. Lähestymistavalle tunnusomaista on ennakkokäsitysten huomioiminen päätöksenteossa. Kullekin luokalle ω_k asetetaan prioritodennäköisyys $P(\omega_k)$ siten, että $\sum_{k=1}^{K} P(\omega_k) = 1$. Ainoastaan ennakkotietoon nojautuen havainto luokitellaan luokkaan, jolle prioritodennäköisyys on suurin, ts. luokkaan ω_k, mikäli $P(\omega_k) > P(\omega_{k'})$ kaikille $k \neq k'$. Käytännön tilanteissa tiedetään kuitenkin enemmän. Luokittelun tehostamiseksi käytetään apuna havainnoista mitattuja muuttuja eli piirteitä. Oletetaan, että kustakin havainnosta on mitattu p piirrettä, ja merkitään havainnon mittauksia vektorilla $\mathbf{x} = (x_1, \ldots, x_p)^{\mathrm{T}}$. Lisäksi määritellään ehdollinen todennäköisyystiheysfunktio $p(\mathbf{x}|\omega_k)$ piirteiden $\mathbf{x}$ arvoille luokan ollessa ω_k. Lausekkeesta $p(\mathbf{x}|\omega_k)$ käytetään usein nimitystä uskottavuus.

Oletetaan tunnetuiksi prioritodennäköisyydet $P(\omega_k) = \pi_k$, $k = 1, \ldots, K$, sekä ehdolliset tiheydet $p(\mathbf{x}|\omega_k)$. Posterioritodennäköisyys luokalle ω_k saadaan laskettua kaavalla

$$P(\omega_k|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_k)P(\omega_k)}{p(\mathbf{x})}, \tag{1}$$

missä

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{x}|\omega_k)P(\omega_k)$$

on skaalaustekijä, joka varmistaa posterioritodennäköisyyksien summautumisen ykköseksi. Posterioritodennäköisyys kuvaa, miten ennakkokäsitys luokasta ω_k muuttuu, kun on mitattu piirteiden $\mathbf{x}$ arvot.

2.1 Päätössääntö

Päätössääntö muodostetaan posterioritodennäköisyyksien perusteella. Havainto luokitellaan luokkaan ω_k, mikäli $P(\omega_k|\mathbf{x}) > P(\omega_{k'}|\mathbf{x})$ kaikille $k \neq k'$. Tämä päätössääntö minimoi luokitteluvirheen todennäköisyyden (Duda et al, 2000): Kahden luokan tapauksessa virheen todennäköisyys ehdolla piirrevektori

$$P(virhe|\mathbf{x}) = \begin{cases} P(\omega_1|\mathbf{x}), & \text{jos valitaan } \omega_2 \\ P(\omega_2|\mathbf{x}), & \text{jos valitaan } \omega_1. \end{cases}$$

Virheen todennäköisyys minimoituu valittaessa ω_1, mikäli $P(\omega_1|\mathbf{x}) > P(\omega_2|\mathbf{x})$ ja vastaavasti ω_2, mikäli $P(\omega_2|\mathbf{x}) > P(\omega_1|\mathbf{x})$. Tällöin virheen todennäköisyys voidaan kirjoittaa muodossa

$$P(virhe|\mathbf{x}) = \min[P(\omega_1|\mathbf{x}), P(\omega_2|\mathbf{x})].$$

Vastaavasti usean luokan tapauksessa virheen todennäköisyys ehdolla piirrevektori

$$P(virhe|\mathbf{x}) = \begin{cases} 1 - P(\omega_1|\mathbf{x}), & \text{jos ei valita } \omega_1 \\ 1 - P(\omega_2|\mathbf{x}), & \text{jos ei valita } \omega_2, \\ \vdots & \\ 1 - P(\omega_K|\mathbf{x}), & \text{jos ei valita } \omega_K. \end{cases}$$

Virheen todennäköisyys minimoituu valittaessa ω_1, mikäli $P(\omega_1|\mathbf{x})$ on suurin ja ω_2, mikäli $P(\omega_2|\mathbf{x})$ on suurin. Yleisemmin: Virheen todennäköisyys minimoituu, kun luokitellaan havainto luokkaan, jonka posterioritodennäköisyys on suurin. Tällöin virheen todennäköisyys voidaan kirjoittaa muodossa

$$P(virhe|\mathbf{x}) = \min[1 - P(\omega_1|\mathbf{x}), 1 - P(\omega_2|\mathbf{x}), \ldots, 1 - P(\omega_k|\mathbf{x})]. \tag{2}$$

Bayesin päätössääntö minimoi keskimääräisen virheen todennäköisyyden, jota kutsutaan Bayes-virheeksi. Se saadaan laskettua kaavalla

$$P(virhe) = \int_{-\infty}^{\infty} P(virhe, \mathbf{x}) d\mathbf{x} = \int_{-\infty}^{\infty} P(virhe|\mathbf{x}) p(\mathbf{x}) d\mathbf{x}. \tag{3}$$

Jos jokaiselle piirteiden $\mathbf{x}$ arvolle $P(virhe|\mathbf{x})$ (2) on mahdollisimman pieni, on integraalista laskettu virheen todennäköisyys (3) myös mahdollisimman pieni.

Tarkastellaan vielä virheen todennäköisyyden (3) laskemista. Kahden luokan tapauksessa piirreavaruus jaetaan kahteen alueeseen $\mathcal{R}_1$ ja $\mathcal{R}_2$, kenties epäoptimaalisesti (Duda et al., 2000). Tällöin virhe voi tapahtua joko siten, että havaittu $\mathbf{x}$ kuuluu alueeseen $\mathcal{R}_2$ todellisen luokan ollessa ω_1 tai alueeseen $\mathcal{R}_1$ todellisen luokan ollessa ω_2. Virheen todennäköisyydeksi saadaan

$$\begin{aligned} P(virhe) &= P(\mathbf{x} \in \mathcal{R}_2, \omega_1) + P(\mathbf{x} \in \mathcal{R}_1, \omega_2) \\ &= P(\mathbf{x} \in \mathcal{R}_2|\omega_1)P(\omega_1) + P(\mathbf{x} \in \mathcal{R}_1|\omega_2)P(\omega_2) \\ &= \int_{\mathcal{R}_2} p(\mathbf{x}|\omega_1)P(\omega_1)d\mathbf{x} + \int_{\mathcal{R}_1} p(\mathbf{x}|\omega_2)P(\omega_2)d\mathbf{x}. \end{aligned} \tag{4}$$

Yleisessä tapauksessa, kun luokkia on useampia, on kätevämpää laskea todennäköisyys olla oikeassa

$$
\begin{aligned}
P(oikein) &= \sum_{k=1}^{K} P(\mathbf{x} \in \mathcal{R}_k, \omega_k) \\
&= \sum_{k=1}^{K} P(\mathbf{x} \in \mathcal{R}_k | \omega_k) P(\omega_k) \\
&= \sum_{k=1}^{K} \int_{\mathcal{R}_k} p(\mathbf{x}|\omega_k) P(\omega_k) d\mathbf{x},
\end{aligned}
$$

jossa $p(\mathbf{x}|\omega_k)P(\omega_k) = p(\omega_k|\mathbf{x})p(\mathbf{x})$ kaavan (1) perusteella. Edellä esitetty tulos ei riipu siitä, miten piirreavaruus on jaettu alueisiin eikä jakauman muodosta (Duda et al., 2000). Havainto luokitellaan Bayesin päätössäännön mukaisesti luokkaan ω_k, mikäli $P(\omega_k|\mathbf{x}) > P(\omega_{k'}|\mathbf{x})$ kaikille $k \neq k'$, jolloin luokitteluvirheen todennäköisyys minimoituu (Duda et al., 2000).

Edellä esiteltiin yksinkertaistettu tilanne, jossa ei otettu huomioon vääristä luokitteluista aiheutuvia tappioita. Toisin sanoen kaikki väärät luokittelupäätökset ajateltiin samanarvoisiksi. Laajennetaan edellä esitettyä teoriaa ottamalla käyttöön tappiofunktion käsite. Olkoot $\omega_1, \ldots, \omega_K$ ennalta tunnetut luokat ja $\alpha_1, \ldots, \alpha_a$ toiminnot. Havainto voidaan myös jättää kokonaan luokittelematta posteriorijakauman perusteella, joten voi olla $a > K$. Määritellään tappiofunktio $\lambda(\alpha_i|\omega_k)$, joka määrittää tappion, kun valitaan toiminto α_i todellisen luokan ollessa ω_k. Valittaessa toiminto α_i ehdollinen odotettu tappio ehdolla piirrevektorin $\mathbf{x}$ arvo

$$
R(\alpha_i|\mathbf{x}) = \sum_{k=1}^{K} \lambda(\alpha_i|\omega_k) P(\omega_k|\mathbf{x}), \tag{5}
$$

jota kutsutaan myös ehdolliseksi riskiksi. Tavoitteena on löytää päätössääntö, joka minimoi kokonaisriskin.

Täsmällisemmin: Olkoon $\alpha(\mathbf{x})$ päätössääntöfunktio, joka määrittää päätöksen kaikille mahdollisille havainnoille, ts. kullekin piirrevektorin $\mathbf{x}$ arvolle määritetään toiminto α_i, $i = 1, \ldots, a$. Kokonaisriski R, jota kutsutaan Bayes-riskiksi, on odotettu tappio valitulle päätössäännölle

$$
R = \int R(\alpha(\mathbf{x})|\mathbf{x}) p(\mathbf{x}) d\mathbf{x}, \tag{6}
$$

joka saadaan integroimalla yli piirreavaruuden. Bayesin päätössäännön mukaan kokonaisriski minimoituu, kun kullekin toiminnolle α_i ehdollinen riski $R(\alpha_i|\mathbf{x})$ on mahdollisimman pieni. Erikoistapaus on symmetrinen 0-1-tappiofunktio

$$
\lambda(\alpha_i|\omega_k) = \begin{cases} 0, & i = k \\ 1, & i \neq k, \end{cases}
$$

jolloin väärät luokittelupäätökset ajatellaan samanarvoisiksi. Tällöin ehdollinen riski vastaa ehdollisen virheen todennäköisyyttä

$$\begin{aligned} R(\alpha_i|\mathbf{x}) &= \sum_{k=1}^{K} \lambda(\alpha_i|\omega_k) P(\omega_k|\mathbf{x}) \\ &= \sum_{k \neq i} P(\omega_k|\mathbf{x}) \\ &= 1 - P(\omega_i|\mathbf{x}). \end{aligned}$$

2.2 Lineaarinen ja kvadraattinen luokittelusääntö

Olkoon $f_k(\mathbf{x}) = p(\mathbf{x}|\omega_k)$ luokan ω_k piirteiden tiheysfunktio ja $\pi_k = P(\omega_k)$ luokkaan ω_k kuulumisen prioritodennäköisyys. Posterioritodennäköisyys luokalle ω_k saadaan laskettua kaavalla

$$P(\omega_k|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_k)P(\omega_k)}{p(\mathbf{x})} = \frac{\pi_k f_k(\mathbf{x})}{\sum_{k'=1}^{K} \pi_{k'} f_{k'}(\mathbf{x})}. \tag{7}$$

Edellä esitetyn Bayes-luokittelijan (7) rakenteen määräävät ehdolliset tiheydet $f_k(\mathbf{x})$ ja prioritodennäköisyydet π_k, joille $\sum_{k=1}^{K} \pi_k = 1$. Prioritodennäköisyydet voidaan olettaa yhtä suuriksi tai esimerkiksi estimoida opetusaineistosta, mikäli opetusaineistoa voidaan pitää satunnaisotoksena populaatiojakaumasta (Friedman, 1989). Mikäli oletetaan, että $f_k(\mathbf{x})$ on normaalijakauman tiheysfunktio, päädytään lineaariseen tai kvadraattiseen luokittelusääntöön.

Seuraavaksi mallinnetaan luokkatiheyksiä p-ulotteisella normaalijakaumalla, jonka tiheysfunktio on muotoa

$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2} d_k^2(\mathbf{x})\right), \tag{8}$$

missä

$$d_k(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu}_k)^{\mathrm{T}} \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)} \tag{9}$$

on Mahalanobis-etäisyys, joka ottaa huomioon luokan ω_k muuttujien kovarianssirakenteen. Johdetaan yhtäpitävät luokittelusäännöt lausekkeesta (7). Jos kaikille $k \neq k'$

$$\begin{aligned} & P(\omega_k|\mathbf{x}) \geq P(\omega_{k'}|\mathbf{x}) \\ \Leftrightarrow\ & \pi_k f_k(\mathbf{x}) \geq \pi_{k'} f_{k'}(\mathbf{x}) \\ \Leftrightarrow\ & -2\ln(\pi_k f_k(\mathbf{x})) \leq -2\ln(\pi_{k'} f_{k'}(\mathbf{x})) \\ \Leftrightarrow\ & -2\ln(\pi_k) + \ln|\boldsymbol{\Sigma}_k| + d_k^2(\mathbf{x}) \leq -2\ln(\pi_{k'}) + \ln|\boldsymbol{\Sigma}_{k'}| + d_{k'}^2(\mathbf{x}), \end{aligned}$$

niin havainto $\mathbf{x}$ luokitellaan luokkaan ω_k. Määritellään vielä viimeisimmän lausekkeen vasemmasta puolesta luokalle k luokittelupistemäärä

$$g_k(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu}_k)^{\mathrm{T}} \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) + \ln |\boldsymbol{\Sigma}_k| - 2 \ln(\pi_k). \tag{10}$$

Siis havainto $\mathbf{x}$ luokitellaan luokkaan ω_k, mikäli $g_k(\mathbf{x}) < g_{k'}(\mathbf{x})$ kaikille $k \neq k'$. Lausekkeesta (10) saadaan kvadraattinen luokittelusääntö (*Quadratic Discriminant Analysis*, lyh. QDA), jolloin aineistoa erottelevat rajat ovat toisen asteen käyriä. Mikäli kovarianssimatriisit $\boldsymbol{\Sigma}_k$ oletetaan yhtä suuriksi, niin päädytään lineaariseen luokitteluun (*Linear Discriminant Analysis*, lyh. LDA), jolloin aineistoa erottelevat rajat ovat suoria.

Tarkastellaan vielä esimerkin avulla, miten päätössääntöjä muodostetaan. Kuvassa 1 on esimerkki päätössäännön muodostamisesta kvadraattisella luokittelijalla (QDA), jolloin päätössääntö huomioi kunkin luokan kovarianssirakenteen. Kuvan ellipsit on piirretty `R`-ohjelmointiympäristön (R Core Team, 2016) `mixtools`-paketin `ellipse`-funktiolla, jolle annetaan parametreina odotusarvovektorit $\boldsymbol{\mu}_k$ ja kovarianssimatriisit $\boldsymbol{\Sigma}_k$. Parametrit oletetaan tunnetuiksi:

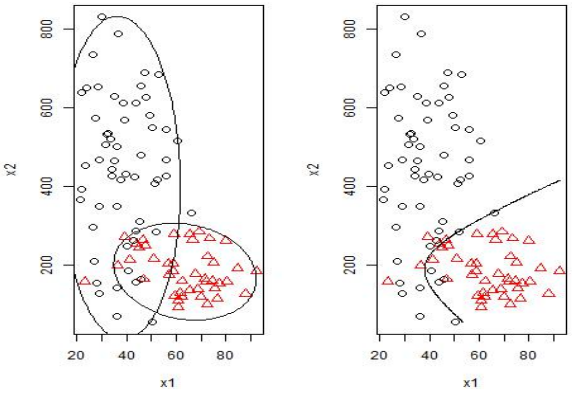
$$\pi_1 = 0.54, \quad \pi_2 = 0.46$$

$$\boldsymbol{\mu}_1 = (37.6, 421.7)^{\mathrm{T}}, \quad \boldsymbol{\mu}_2 = (63.5, 183.0)^{\mathrm{T}}$$

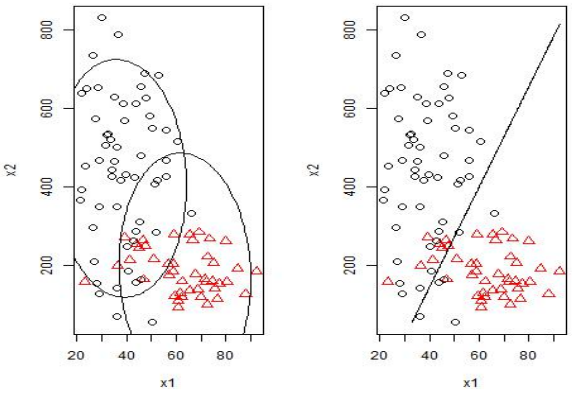
$$\boldsymbol{\Sigma}_1 = \begin{pmatrix} 96.6 & -83.4 \\ -83.4 & 28174.9 \end{pmatrix}, \quad \boldsymbol{\Sigma}_2 = \begin{pmatrix} 137.4 & -90.5 \\ -90.5 & 2537.7 \end{pmatrix}.$$

Kunkin luokan ellipsi piirretään siten, että kyseisen luokan kaksiulotteisesta normaalijakaumasta peräisin oleva havainto kuuluu 95 prosentin todennäköisyydellä ellipsin sisälle. Ellipsin kehällä olevat pisteet $\mathbf{x}$ ovat yhtä kaukana Mahalanobis-etäisyyden (9) mielessä kyseisen luokan odotusarvovektorista $\boldsymbol{\mu}_k$. Ellipsejä voisi piirtää myös muilla Mahalanobis-etäisyyksillä. Vastaavasti kuvassa 2 on esimerkki päätössäännön muodostamisesta lineaarisella luokittelijalla (LDA), jolloin kovarianssimatriisit yhdistetään, ts. ne ovat samat kullekin luokalle.

Päätössääntö QDA:lle ja LDA:lle voidaan piirtää käyttäen `R`-ohjelmointiympäristön `contour`-funktiota. Tätä varten ensin luodaan 500×500 hila, siten että x-akselin ja y-akselin rajoiksi asetetaan piirteiden minimi- ja maksimiarvot. Kullekin hilan pisteelle ennustetaan luokka käyttäen lineaarista tai kvadraattista luokittelua (Kuva 1 ja 2, oikea kuva). R-koodi päätössäännön piirtämiseen (ks. Liite C) on esitetty sivustolla `http://www.cbcb.umd.edu/~hcorrada/PracticalML/src/classification.R`.



Kuva 1: QDA:n päätössääntö simuloidulle aineistolle, jossa on kaksi luokkaa ja kaksi piirrettä. Vasemmassa kuvassa on Mahalanobis-etäisyydet kummallekin luokalle siten, että havainnot kuuluvat 95 prosentin todennäköisyydellä ellipsien sisälle. Oikeassa kuvassa on päätössääntö kvadraattiselle luokittelijalle (QDA), joka saadaan asettamalla $g_1(\mathbf{x}) = g_2(\mathbf{x})$, jolloin aineistoa erotteleva raja on toisen asteen käyrä. Luokittelupistemäärän (10) parametrit oletetaan tunnetuiksi.



Kuva 2: LDA:n päätössääntö simuloidulle aineistolle, jossa on kaksi luokkaa ja kaksi piirrettä. Vasemmassa kuvassa on luokkien kovarianssirakennetta kuvaavat ellipsit lineaariselle luokittelijalle (LDA). Oikeassa kuvassa on päätössääntö lineaariselle luokittelijalle (LDA), joka saadaan asettamalla $g_1(\mathbf{x}) = g_2(\mathbf{x})$, jolloin aineistoa erotteleva raja on suora. Luokittelupistemäärän (10) parametrit oletetaan tunnetuiksi.

Edellä konstruoitiin luokittelijoita olettaen prioritodennäköisyydet $P(\omega_k)$ ja ehdolliset tiheydet $p(\mathbf{x}|\omega_k)$ tunnetuiksi. Käytännössä lausekkeen (10) parametrit ovat kuitenkin tuntemattomia, joten ne korvataan opetusaineistosta estimoiduilla parametreilla. Estimointimenetelmänä voidaan käyttää esimerkiksi suurimman uskottavuuden menetelmää, ks. Duda et al. (2000) tai momenttimenetelmää, joka on oletusasetuksena `R`-ohjelmointiympäristön `MASS`-paketin `lda`- ja `qda`-funktiolla. Olkoot N yksilöiden lukumäärä ja N_k luokan ω_k yksilöiden lukumäärä opetusaineistossa. Silloin suurimman uskottavuuden estimaatit voidaan esittää seuraavasti

$$\hat{\pi}_k = \frac{N_k}{N}, \tag{11}$$

$$\hat{\boldsymbol{\mu}}_k = \bar{\mathbf{x}}_k = \frac{1}{N_k} \sum_{c(i)=k} \mathbf{x}_i, \tag{12}$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{1}{N_k} \sum_{c(i)=k} (\mathbf{x}_i - \bar{\mathbf{x}}_k)(\mathbf{x}_i - \bar{\mathbf{x}}_k)^{\mathrm{T}}, \tag{13}$$

jossa $c(i) = k$ tarkoittaa, että yksilön i luokka on ω_k. Lineaarisen luokittelun tapauksessa estimoidaan yhdistetty kovarianssimatriisi

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N - K} \sum_{k=1}^{K} \sum_{c(i)=k} (\mathbf{x}_i - \bar{\mathbf{x}}_k)(\mathbf{x}_i - \bar{\mathbf{x}}_k)^{\mathrm{T}}. \tag{14}$$

Friedmanin (1989) mukaan lineaarisen ja kvadraattisen luokittelun voidaan olettaa toimivan hyvin, mikäli luokkien tiheydet ovat normaalisia ja saadaan laskettua erottelumielessä hyvät estimaatit parametreille $\boldsymbol{\mu}_k$ ja $\boldsymbol{\Sigma}_k$. Lineaarinen ja kvadraattinen luokittelija ovat käyttökelpoisia työkaluja monenlaisiin ongelmiin, mistä osoituksena on niiden menestyminen luokittelumenetelmien vertailussa STATLOG-projektissa (Michie et al., 1994). Menetelmien tehon voidaan olettaa johtuvan siitä, että usein aineisto tukee ainoastaan yksinkertaisia erottelevia rajoja (Hastie et al., 2009).

Ongelmia tulee, kun luokkakoot N_k ovat pieniä verrattuna piirreavaruuden dimensioon p. Tällöin kovarianssimatriisien estimoiminen muuttuu epätarkemmaksi, ja kaikki parametrit eivät välttämättä ole edes identifioituvia. Vaikutuksen voi huomata johtamalla kovarianssimatriisille spektraalihajotelman

$$\boldsymbol{\Sigma}_k = \sum_{j=1}^{p} e_{jk} \mathbf{v}_{jk} \mathbf{v}_{jk}^{\mathrm{T}}, \tag{15}$$

missä e_{jk} on kovarianssimatriisin $\boldsymbol{\Sigma}_k$ j:s ominaisarvo ja $\mathbf{v}_{jk}$ sitä vastaava ominaisvektori (Friedman, 1989). Kovarianssimatriisin $\boldsymbol{\Sigma}_k$ käänteismatriisi näiden avulla esitettynä on

$$\boldsymbol{\Sigma}_k^{-1} = \sum_{j=1}^{p} \frac{\mathbf{v}_{jk} \mathbf{v}_{jk}^{\mathrm{T}}}{e_{jk}}, \tag{16}$$

jolloin kaavan (10) luokittelupistemäärä $g_k(\mathbf{x})$ voidaan esittää muodossa

$$g_k(\mathbf{x}) = \sum_{j=1}^{p} \frac{[\mathbf{v}_{jk}^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu}_k)]^2}{e_{jk}} + \sum_{j=1}^{p} \ln e_{jk} - 2 \ln \pi_k. \tag{17}$$

Lausekkeesta (17) näkee, että pienimmillä ominaisarvoilla on suuri painoarvo luokittelupistemäärään. Kovarianssiestimaatit (13) tuottavat harhaisia estimaatteja ominaisarvoille (Friedman, 1989). Kun $N_k \leq p$, pienimmät ominaisarvot estimoituvat

nolliksi. Tällöin luokan ω_k otoskovarianssimatriisi ei ole kääntyvä, jolloin ei pystytä laskemaan luokittelupistemäärää (17) luokalle ω_k. Ongelmaa voidaan korjata regularisointimenetelmillä, joiden tarkoituksena on säädellä mallin kompleksisuutta. Tästä esimerkkinä on seuraavaksi esiteltävä Friedmanin (1989) klassinen regularisointi. Toinen lähestymistapa on piirteiden vähentäminen, jolloin myös estimoitavien parametrien määrä vähenee.

2.2.1 Klassinen regularisointi

Kun havaintoja on vähän suhteessa piirteiden määrään, mallin parametreja ei pystytä estimoimaan stabiilisti. Regularisoinnin ideana on työntää parametriestimaatteja kohti arvoja, joita voidaan pitää fysikaalisista syistä järkevinä. Regularisoinnin tavoitteena on pienentää varianssia lisäämällä pientä harhaa tuloksiin. Friedman (1989) esitti regularisointimenetelmän (*Regularized Discriminant Analysis*, lyh. RDA), joka on välimuoto LDA:sta ja QDA:sta.

Otetaan käyttöön merkintä

$$\hat{\mathbf{\Sigma}}_k(\lambda) = (1-\lambda)\hat{\mathbf{\Sigma}}_k + \lambda\hat{\mathbf{\Sigma}}. \tag{18}$$

Edellä tavoitteena on kutistaa kovarianssimatriiseja $\hat{\mathbf{\Sigma}}_k$ kohti yhdistettyä kovarianssimatriisia $\hat{\mathbf{\Sigma}}$. Regularisointiparametri λ kuvaa kutistamisen astetta ja saa arvoja väliltä 0 ja 1. Arvolla $\lambda = 0$ saadaan $\hat{\mathbf{\Sigma}}_k(\lambda) = \hat{\mathbf{\Sigma}}_k$, jolloin päädytään kvadraattiseen luokitteluun (QDA), ja vastaavasti arvolla $\lambda = 1$ saadaan $\hat{\mathbf{\Sigma}}_k(\lambda) = \hat{\mathbf{\Sigma}}$, jolloin päädytään lineaariseen luokitteluun (LDA).

Kovarianssimatriisien kutistaminen kohti yhdistettyä kovarianssimatriisia ei välttämättä ole tehokkainta. Mikäli kovarianssimatriisit ovat yksikkömatriisin monikertoja, niin järkevämpää on kutistaa kovarianssimatriiseja kohti yksikkömatriisia $\mathbf{I}$ kerrottuna keskimääräisellä ominaisarvollaan. Lauseke (18) voidaan päivittää muotoon

$$\hat{\mathbf{\Sigma}}_k(\lambda,\gamma) = (1-\gamma)\hat{\mathbf{\Sigma}}_k(\lambda) + \frac{\gamma}{p}\operatorname{tr}[\hat{\mathbf{\Sigma}}_k(\lambda)]\mathbf{I}, \tag{19}$$

missä $\hat{\mathbf{\Sigma}}_k(\lambda)$ määritellään lausekkeen (18) mukaisesti, ja γ on toinen regularisointiparametri, joka kuvaa kutistamista kohti yksikkömatriisin monikertaa. Kutistaminen pienentää suuria odotusarvoja ja suurentaa pieniä ominaisarvoja. Päädytään luokittelupistemäärään

$$g_k(\mathbf{x}) = (\mathbf{x}-\bar{\mathbf{x}}_k)^{\mathrm{T}}\hat{\mathbf{\Sigma}}_k^{-1}(\lambda,\gamma)(\mathbf{x}-\bar{\mathbf{x}}_k) + \ln|\hat{\mathbf{\Sigma}}_k(\lambda,\gamma)| - 2\ln\pi_k. \tag{20}$$

Käytännössä parametrien λ ja γ arvoja ei tunneta, vaan ne estimoidaan opetusaineistosta. Tavoitteena on minimoida väärin luokiteltujen havaintojen osuus käymällä läpi tasaväliseltä hilalta useita eri (λ,γ) -arvoja, joista valitaan se parametripari, jolle väärin luokiteltujen havaintojen osuus on pienin. Proseduuriin voidaan soveltaa otoksen uudelleenkäyttömenetelmiä kuten bootsrap-menetelmiä tai ristiinvalidointia, joista kerrotaan tarkemmin luvussa 3. Tässä tutkielmassa regularisointimenetelmää ei kuitenkaan toteuteta käytännössä.

2.2.2 Piirteiden valinta

Piirteiden valinta on yleinen ongelma tilastollisessa hahmontunnistuksessa. Piirteiden määrän vähentämiseen on monia syitä, joista ilmeinen on luokittelutarkkuuden parantaminen ja/tai menetelmän toimivuuden takaaminen. Malliin halutaan piirteitä, joilla on hyvä erottelukyky. Piirteitä voidaan myös yhdistellä esimerkiksi pääkomponenttianalyysin tai erotteluanalyysin avulla.

Muotoillaan lähtötilanne artikkelin Jain & Zongker (1997) pohjalta. Olkoon $\mathbf{x}$ alkuperäisten piirteiden joukko, joka sisältää p alkiota. Olkoon $\mathbf{x}^\star$ alkuperäisten piirteiden osajoukko sisältäen yhteensä d piirrettä. Joukko $\mathbf{x}^\star$ valitaan käyttämällä kriteerifunktiota $J(\mathbf{x}^\star)$, jonka arvo maksimoidaan. Tällöin suuret J:n arvot indikoivat hyviä muuttujajoukkoja. Etsitään siis piirteiden joukko $\mathbf{x}^\star \subseteq \mathbf{x}$, jolle

$$J(\mathbf{x}^\star) = \max J(\mathbf{x}^\star).$$

Luonnollinen valinta kriteerifunktioksi on oikein luokiteltujen havaintojen osuus.

Yksinkertainen ratkaisu olisi käydä läpi kaikki kombinaatiot $\binom{p}{d}$, mutta se ei useinkaan ole mahdollista eikä tarpeen. Piirteiden valinta voidaan ajatella optimointiongelmana, jonka ratkaisemiseksi on tarjolla useita algoritmeja. Jain & Zongker (1997) esittävät, että algoritmit piirteiden valitsemiselle voidaan ensinnäkin jakaa tilastolliseen hahmontunnistukseen perustuviin menetelmiin ja neuroverkkoja hyödyntäviin menetelmiin. Vastaavasti tilastolliseen hahmontunnistukseen perustuvat menetelmät voidaan jakaa optimaalisiin ja suboptimaalisiin menetelmiin. Yksinkertaisin esimerkki optimaalisesta menetelmästä on tyhjentävä etsintä, joka käy läpi kaikki 2^p piirteiden osajoukkoa. On kuitenkin olemassa muitakin algoritmeja, jotka takaavat optimaalisen tuloksen. Suboptimaaliset algoritmit ovat paljon yleisempiä. Ne perustuvat jonkinlaiseen heuristiikkaan, jonka avulla muuttujien valinnasta suoriudutaan paljon lyhyemmässä ajassa. Optimaalinen ratkaisu voi tosin jäädä löytämättä. Suboptimaaliset algoritmit voidaan jakaa deterministisiin algoritmeihin, jotka löytävät aina saman muuttujajoukon, ja stokastisiin algoritmeihin, jotka löytävät erilaisia muuttujajoukkoja eri suorituskerroilla.

Yksinkertainen esimerkki deterministisistä algoritmeista on askeltava algoritmi. Se toimii siten, että malliin joko lisätään tai mallista poistetaan yksitellen piirre halutun kriteerin mukaan. Eteenpäin askeltavassa versiossa lähdetään liikkeelle tyhjästä piirrejoukosta. Ensimmäisenä malliin lisätään se piirre, joka yksinään luokittelee yksilöt parhaiten. Seuraavilla kierroksilla malliin lisätään yksitellen ne piirteet, joita ei ole vielä mallissa mukana. Näin jatketaan kunnes saavutetaan asetettu lopettamiskriteeri. Algoritmi voidaan esimerkiksi lopettaa, kun oikein luokiteltujen havaintojen osuus ei olennaisesti kasva. Taaksepäin askeltavassa versiossa lähdetään liikkelle mallista, joka sisältää kaikki piirteet. Mallista pudotetaan pois yksitellen kukin piirre, kunnes saavutetaan paras luokittelutulos.

Tässä tutkielmassa piirteiden valinta on osa luvun 4 simulointikoetta. Piirteiden vähentämiseen käytetään askeltavaa algoritmia QDA:n toiminnan takaamiseksi, muutoin valintamenetelmät rajataan tämän työn ulkopuolelle. Piirteet valitaan lisäämällä tyhjään piirrejoukkoon yksitellen haluttu määrä piirteitä käyttäen `R`-ohjelmointiympäristön `klaR`-paketin `stepclass`-funktiota, joka toteuttaa piirteiden valinnan

käyttäen luvussa 3 esiteltyä ristiinvalidointia. On syytä huomata, että tuloksena saatu piirteiden joukko riippuu käytettävästä luokittelumenetelmästä sekä opetus- ja testiaineiston koosta. Eri iterointikerroilla voidaan myös saada hyvinkin erilaisia piirrejoukkoja johtuen opetus- ja testiaineistojakoon liittyvästä satunnaisuudesta.

3 Bayes-virheen määräämisestä

Bayes-virheen eli virheen todennäköisyyden (3) laskeminen muodostuu hankalaksi jo toisessa dimensiossa. Tässä luvussa esitetään erilaisia tapoja arvioida Bayes-virhettä.

3.1 Chernoffin yläraja

Kahden luokan tapauksessa virheen todennäköisyyden (3) ylärajaa voidaan arvioida analyyttisesti. Lasketaan Chernoffin yläraja virheen todennäköisyydelle (Duda et al., 2000). Apuna käytetään epäyhtälöä

$$\min[a,b] \leq a^{\beta} b^{1-\beta} \qquad \text{kaikille} \qquad a,b \geq 0 \qquad \text{ja} \qquad 0 \leq \beta \leq 1. \tag{21}$$

Soveltamalla edellä esitettyä epäyhtälöä (21) virheen todennäköisyyden lausekkeisiin (2) ja (3) saadaan epäyhtälö

$$\begin{aligned}
P(virhe) &= \int P(virhe|\mathbf{x})p(\mathbf{x})d\mathbf{x} = \int \min[P(\omega_1|\mathbf{x}), P(\omega_2|\mathbf{x})]p(\mathbf{x})d\mathbf{x} \\
&\leq \int P^{\beta}(\omega_1|\mathbf{x})P^{1-\beta}(\omega_2|\mathbf{x})p(\mathbf{x})d\mathbf{x} \\
&= \int \left[\frac{p(\mathbf{x}|\omega_1)P(\omega_1)}{p(\mathbf{x})}\right]^{\beta} \left[\frac{p(\mathbf{x}|\omega_2)P(\omega_2)}{p(\mathbf{x})}\right]^{1-\beta} p(\mathbf{x})d\mathbf{x} \\
&= \int p^{\beta}(\mathbf{x}|\omega_1)P^{\beta}(\omega_1)p^{1-\beta}(\mathbf{x}|\omega_2)P^{1-\beta}(\omega_2)p(\mathbf{x})^{-\beta}p(\mathbf{x})^{\beta-1}p(\mathbf{x})d\mathbf{x} \\
&= P^{\beta}(\omega_1)P^{1-\beta}(\omega_2)\int p^{\beta}(\mathbf{x}|\omega_1)p^{1-\beta}(\mathbf{x}|\omega_2)d\mathbf{x}, \qquad 0 \leq \beta \leq 1,
\end{aligned} \tag{22}$$

jossa integrointialueena on koko piirreavaruus. Mikäli voidaan olettaa, että ehdolliset tiheydet noudattavat normaalijakaumaa, saadaan epäyhtälö määrättyä sijoittamalla normaalijakauman ehdolliset tiheydet $p(\mathbf{x}|\omega_1)$ ja $p(\mathbf{x}|\omega_2)$ (8) yhtälöön (22). Integraaliosa voidaan esittää tällöin analyyttisesti

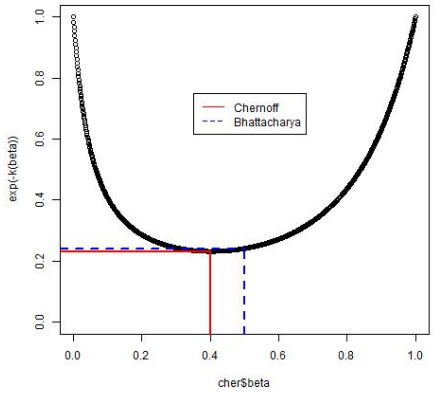
$$\int p^{\beta}(\mathbf{x}|\omega_1)p^{1-\beta}(\mathbf{x}|\omega_2)d\mathbf{x} = e^{-k(\beta)}, \tag{23}$$

missä

$$k(\beta) = \frac{1}{2}\beta(1-\beta)(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^{\mathrm{T}}[\beta\boldsymbol{\Sigma}_1+(1-\beta)\boldsymbol{\Sigma}_2]^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)+\frac{1}{2}\ln\left(\frac{|\beta\boldsymbol{\Sigma}_1+(1-\beta)\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|^{\beta}|\boldsymbol{\Sigma}_2|^{1-\beta}}\right)$$

(Duda et al., 2000). Tarkoituksena on löytää β, joka minimoi funktion $e^{-k(\beta)}$. Lopuksi saatu β sijoitetaan yhtälöön (22), jolloin saadaan Chernoffin yläraja virheen todennäköisyydelle. Asettamalla $\beta = 1/2$ saadaan Bhattacharyyan yläraja.

Havainnollistetaan ylärajojen laskemista vielä esimerkin avulla. Määrätään Chernoffin yläraja (22) sekä Bhattacharyyan yläraja. Kuvaan 3 on piirretty funktion $e^{-k(\beta)}$ (23) arvoja eri β:n arvoilla jakamalla väli $[0, 1]$ tasapituisiin väleihin. Funktion $k(\beta)$ parametrit esitettiin s. 7. Lopuksi funktion $e^{-k(\beta)}$ (23) minimoiva β:n arvo sijoitetaan yhtälöön (22), jolloin Chernoffin ylärajaksi saadaan 0.1137, kun $\beta = 0.402$. Vastaavasti Bhattacharyyan ylärajaksi saadaan 0.1203, kun $\beta = 0.5$.



Kuva 3: Funktion $e^{-k(\beta)}$ arvoja eri β:n arvoilla. Chernoffin raja on kohdassa $\beta = 0.402$ ja Bhattacharyyan raja kohdassa $\beta = 0.5$. Chernoffin yläraja on 0.1137 ja Bhattacharyyan yläraja 0.1203.

3.2 Monte Carlo -integrointi

Bayes-virheen (3) laskenta analyyttisesti on hankalaa piirreavaruuden dimension ja luokkien määrän ollessa suuri. Tässä tutkielmassa ratkaisuksi ehdotetaan Monte Carlo -integrointia. Monte Carlo -menetelmillä viitataan yleisesti joukkoon laskennallisia algoritmeja, jotka perustuvat satunnaislukujen simulointiin. Monte Carlo -integroinnilla viitataan tekniikkaan, jolla integraali lasketaan simuloimalla satunnaislukuja. Kyseessä on ei-deterministinen menetelmä, jolla arvioidaan determinististä suuretta. Kullakin realisaatiolla saadaan eri arvo, ja lopputulos on approksimaatio todellisesta arvosta. Numeerisen integroinnin ongelmiin voidaan soveltaa myös useita numeerisia menetelmiä, ks. esim. Evans (1993). Seuraavaksi esitetään Monte Carlo -integroinnin perusidea käyttäen teoksen Robert & Casella (2009) merkintöjä.

Muotoillaan integrointiongelma yleisesti määrittelemättä jakaumaa tarkemmin. Olkoon X jatkuva satunnaismuuttuja, jonka tiheysfunktio on f, ja h muunnosfunktio. Integrointialueena on otosavaruus $\mathcal{X}$. Kiinnostuksen kohteena on satunnaismuuttujan X muunnoksen $h(X)$ odotusarvo, joka voidaan kirjoittaa integraalina

$$\mu = E_X[h(X)] = \int_{\mathcal{X}} h(x)f(x)dx. \tag{24}$$

Sitä voidaan arvioida riippumattomalla otoksella $(X_1, \ldots, X_m)$, joka on generoitu satunnaismuuttujan X jakaumasta. Suurten lukujen lain mukaan otoksesta $(X_1, \ldots, X_m)$

laskettu empiirinen keskiarvo

$$\hat{\mu} = \bar{h}_m = \frac{1}{m} \sum_{j=1}^{m} h(x_j) \tag{25}$$

konvergoi melkein varmasti kohti odotusarvoa $E_X[h(X)]$. Estimaattorin varianssi on

$$\begin{aligned} var(\bar{h}_m) &= var\left[\frac{1}{m}\sum_{j=1}^{m} h(X_j)\right] = \frac{m \cdot var(h(X))}{m^2} = \frac{var(h(X))}{m} \\ &= \frac{1}{m}\int_{\mathcal{X}} (h(x)-\mu)^2 f(x)dx, \end{aligned}$$

jossa X noudattaa samaa jakaumaa kuin otos $(X_1, \ldots, X_m)$. Varianssia $var(\bar{h}_m)$ voidaan arvioida otoksesta $(X_1, \ldots, X_m)$ estimoidulla otosvarianssilla

$$v_m = \hat{var}(\bar{h}_m) = \frac{\frac{1}{m}\sum_{j=1}^{m}(h(x_j)-\bar{h}_m)^2}{m} = \frac{1}{m^2}\sum_{j=1}^{m}(h(x_j)-\bar{h}_m)^2.$$

3.2.1 Bayes-virheen Monte Carlo -integrointi

Tarkastellaan seuraavaksi, miten Monte Carlo -integrointia voidaan hyödyntää Bayes-virheen (3) laskennassa. Palataan vielä lukuun 2, jossa esitettiin virheen todennäköisyyden $P(virhe)$ (4) laskentaa kahden luokan ja yhden piirteen tapauksessa, kun integrointialueet ovat $\mathcal{R}_1$ ja $\mathcal{R}_2$ kuten sivulla 4. Valittaessa toiminto α_1 havainto luokitellaan luokkaan ω_1 ja valittaessa toiminto α_2 havainto luokitellaan luokkaan ω_2. Virheen todennäköisyys (4) voidaan tällöin esittää päätössääntöfunktion $\alpha(x)$ ja 0-1-tappiofunktion $\lambda(\alpha(x)|\omega_k)$, $k = 1, 2$, avulla seuraavasti:

$$\begin{aligned} P(virhe) &= \int_{\mathcal{R}_1} P(\omega_2)p(x|\omega_2)dx + \int_{\mathcal{R}_2} P(\omega_1)p(x|\omega_1)dx \\ &= \int_{\mathcal{R}_1} \lambda(\alpha(x)|\omega_2)P(\omega_2)p(x|\omega_2)dx + \int_{\mathcal{R}_2} \lambda(\alpha(x)|\omega_1)P(\omega_1)p(x|\omega_1)dx, \end{aligned} \tag{26}$$

sillä $\alpha(x) = \alpha_1$, kun $x \in \mathcal{R}_1$, jolloin $\lambda(\alpha(x)|\omega_2) = 1$, muutoin 0. Samoin $\alpha(x) = \alpha_2$, kun $x \in \mathcal{R}_2$, jolloin $\lambda(\alpha(x)|\omega_1) = 1$, muutoin 0. Modifioimalla kaavaa (26) siten, että integroidaan koko reaaliakselin $\mathbb{R}$ yli, Bayes-virheeksi saadaan

$$\begin{aligned} P(virhe) &= \int_{\mathbb{R}} \lambda(\alpha(x)|\omega_2)P(\omega_2)p(x|\omega_2)dx + \int_{\mathbb{R}} \lambda(\alpha(x)|\omega_1)P(\omega_1)p(x|\omega_1)dx \\ &= \int_{\mathbb{R}} \sum_{k=1}^{2} \underbrace{\lambda(\alpha(x)|\omega_k)}_{h(x,\omega_k)} \underbrace{P(\omega_k)p(x|\omega_k)}_{p(x,\omega_k)} dx = E_{x,\omega}[\lambda(\alpha(x|\omega))], \end{aligned} \tag{27}$$

joka on rakenteeltaan samanlainen kuin kaava (24). Tällöin Bayes-virhettä voidaan arvioida simuloimalla m havaintoa yhteisjakaumasta $p(x, \omega_k) = P(\omega_k)p(x|\omega_k)$, joka vastaa yleisen esitystavan funktiota $f(x)$. Vastaavasti $h(x, \omega_k) = \lambda(\alpha(x)|\omega_k)$ vastaa yleisen esitystavan funktiota $h(x)$. Korvataan integraali otoksesta lasketulla empiirisellä keskiarvolla kaavan (25) mukaisesti

$$\hat{\mu} = \bar{h}_m = \frac{1}{m}\sum_{j=1}^{m} h(x_j, \omega_{k_j}) = \frac{1}{m}\sum_{j=1}^{m} \lambda(\alpha(x_j)|\omega_{k_j}), \tag{28}$$

missä m on simuloitujen havaintojen lukumäärä, $\alpha(x_j)$ tunnettu tai estimoitu päättelysääntöfunktio ja ω_{k_j} yksilön j luokka. Funktio $\lambda(\alpha(x_j)|\omega_{k_j})$ saa arvon 1, mikäli havainto luokitellaan väärin eli $\alpha(x_j) = \alpha_{ij} \neq \omega_{kj}$, ts. $\alpha_i \neq \omega_k$ yksilölle j, ts. toiminto on eri kuin luokka. Menetelmä voidaan yleistää usealle luokalle sekä piirteiden määrälle p seuraavasti:

$$P(virhe) = \int_{\mathbb{R}} \cdots \int_{\mathbb{R}} \sum_{k=1}^{K} \underbrace{\lambda(\alpha(\mathbf{x})|\omega_k)}_{h(\mathbf{x},\omega_k)} \underbrace{P(\omega_k)p(\mathbf{x}|\omega_k)}_{p(\mathbf{x},\omega_k)} d\mathbf{x}_1 \cdots d\mathbf{x}_p, \tag{29}$$

jossa $\mathbf{x}$ on p-ulotteinen vektori eli $\mathbf{x} \in \mathbb{R}^p$ ja K on luokkien määrä.

3.3 Arviointimenetelmät

Tämä luku keskittyy Bayes-virheen (3) arviointiin aineistosta, kun käytössä on äärellinen, pienehkö otos. Mallin hyvyyttä voidaan arvioida sen kyvyllä ennustaa riippumatonta testiaineistoa. Pääasiallisena lähteenä käytetään kirjaa Hastie et al. (2009) siten, että sen merkinnät asetetaan vastaamaan luvun 2 merkintöjä.

Rajoitutaan tilanteeseen, jossa vastemuuttuja on kategorinen. Olkoon opetusaineisto $\mathcal{O} = ((\omega_{k_1}, \mathbf{x}_1), \ldots, (\omega_{k_N}, \mathbf{x}_N))$ kokoa N. Tämän työn merkinnöillä opetusaineistosta estimoitu ennustusmalli eli päätössääntö $\hat{\alpha}(\mathbf{x})$ sijoitetaan tappiofunktion lausekkeeseen $\lambda(\hat{\alpha}(\mathbf{x})|\omega))$, joka 0-1-tappiofunktion tapauksessa voidaan esittää indikaattorifunktion avulla muodossa $\mathbb{1}(\hat{\alpha}(\mathbf{x}) \neq \omega)$. Tällöin funktio saa arvon 1, mikäli havainto luokitellaan väärin ja muutoin 0. Päätössääntöfunktio $\hat{\alpha}(\mathbf{x})$ estimoidaan opetusaineistosta. Tällä säännöllä kullekin havainnolle voidaan estimoida toiminto α_i. Testivirhe, josta käytetään usein nimitystä yleistämisvirhe, määritellään odotettuna ennustevirheenä riippumattomasta testiaineistosta kiinteälle opetusaineistolle $\mathcal{O}$ seuraavasti

$$Err_{\mathcal{O}} = E_{\mathbf{x},\omega|\mathcal{O}}[\lambda(\hat{\alpha}(\mathbf{x})|\omega)|\mathcal{O}]. \tag{30}$$

Toisaalta odotettu testivirhe

$$\begin{aligned} Err &= E_{\mathbf{x},\omega,\mathcal{O}}[\lambda(\hat{\alpha}(\mathbf{x})|\omega)] \\ &= E_{\mathcal{O}}[Err_{\mathcal{O}}] \\ &= E_{\mathcal{O}}[E_{\mathbf{x},\omega|\mathcal{O}}[\lambda(\hat{\alpha}(\mathbf{x})|\omega)|\mathcal{O}]], \end{aligned} \tag{31}$$

joka myös riippuu opetusaineen koosta N. Edellä odotusarvo keskiarvoistaa kaiken satunnaisen yli, ts. se huomioi myös opetusaineistosta estimoidun päätössäännön $\hat{\alpha}(\mathbf{x})$, kun taas aikaisemmin kaavassa (27) päätössääntö oletettiin tunnetuksi. Mitä isompi opetusaineisto on, sitä lähempänä $Err_{\mathcal{O}}$ (30) ja Err (31) ovat Bayes-virhettä $P(virhe)$ (28), mikäli $\hat{\alpha}(\mathbf{x})$ on tarkentuva. Ongelma on, että aineisto on äärellinen ja usein pieni.

Eräs tapa arvioida estimaattorin tarkkuutta on estimoida sen opetusvirhe, joka määritellään opetusaineiston keskimääräisenä tappiona

$$\overline{err} = \frac{1}{N} \sum_{j=1}^{N} \lambda(\alpha(\mathbf{x}_j|\omega_{k_j})).$$

Tällöin opetusaineistoa käytetään sekä luokittelusäännön muodostamiseen että luokitteluvirheen todennäköisyyden estimointiin. Opetusvirhe on kuitenkin huono mittari yleistettävyydelle, joka kuvaa menetelmän kykyä ennustaa riippumatonta testiaineistoa, ts. luokittelijan pitäisi toimia hyvin myös samasta populaatiosta poimittujen uusien riippumattomien otosten tapauksessa. Tästä syystä kiinnostuksen kohteena on estimoidun päätössäännön $\hat{\alpha}(\mathbf{x})$ odotettu testivirhe.

3.3.1 Jako opetus- ja testiaineistoksi

Mikäli aineisto on suuri, niin Hastie et al. (2009) esittävät, että aineisto olisi hyvä jakaa satunnaisesti kolmeen osaan: opetus-, validointi- ja testiaineistoksi. Tällöin testiaineisto tulisi ottaa jo alusta lähtien erilleen harhan välttämiseksi. Opetusaineistoa käytetään mallin sovittamiseen ja validointiaineistoa kyseisen mallin ennustevirheen estimoimiseen. Lopuksi testiaineistolla estimoidaan yleistämisvirhe $Err_{\mathcal{O}}$ käyttäen lopullista "parasta" mallia. Aineiston jakamiselle eri osiin ei voida antaa selkeää sääntöä. Hastie et al. (2009) mukaan tyypillinen jako voisi olla puolet havainnoista opetusaineistoksi ja 1/4 havainnoista validointi- ja testiaineistoksi.

Usein on kuitenkin tilanne, että havaintoja on liian vähän jaettavaksi kolmeen osaan. Tällöin voidaan ottaa käyttöön otoksen uudelleenkäyttömenetelmiä, jotka toimivat ilman validointidataa. Menetelmien etuna on, että ne estimoivat suoraan testivirhettä Err (31). Eräs menetelmä on aineiston jakaminen satunnaisesti opetus- ja testiaineistoksi. Menetelmään törmää usein kirjallisuudessa, jossa siitä käytetään mm. nimityksiä *learning-test split* ja *hold-out method* (McLachlan, 1992). Jakosuhde opetus- ja testiaineistoksi voidaan valita mielivaltaisesti, mutta varsinkin kompleksisille malleille olisi järkevää valita suurempi osa aineistosta opetusaineistoksi. Kirjallisuudessa näkee käytettävän paljon jakosuhteita 50/50, 70/30, 80/20 ja 90/10. Tehtäessä jako kerran kukin havainto voi kuulua vain joko opetus- tai testiaineistoon, mistä voi aiheutua harhaa tuloksiin (Molinaro et al., 2005). Myös opetusaineiston pienuus voi aiheuttaa harhaa tuloksiin, sillä mallin parametreja ei välttämättä pystytä estimoimaan luotettavasti. Tyypillisesti jako opetus- ja testiaineistoksi suoritetaankin useaan kertaan – esimerkiksi 100 kertaa, jolloin estimoitujen ennustevirheiden keskiarvosta saadaan estimaatti yleistämisvirheelle Err. Kirjallisuudessa menetelmästä käytetään mm. nimityksiä *repeated hold-out method* ja *Monte Carlo*

cross-validation (Borra & Di Ciaccio, 2010). Menetelmän selkeä etu on laskennallinen helppous.

3.3.2 Ristiinvalidointi

Otoksen uudelleenkäyttömenetelmistä käytetyin lienee ristiinvalidointi, joka myös estimoi suoraan testivirhettä Err ilman validointidataa. K-kertaisessa ristiinvalidoinnissa aineisto arvotaan suunnilleen K:hon yhtä suureen osaan. Ensin sovitetaan malli $K-1$ osalle, jonka jälkeen lasketaan ennustevirhe pois jätetylle osalle. Toimenpide suoritetaan K kertaa siten, että kullekin osalle vuorollaan estimoidaan ennustevirhe, ja lopuksi yhdistetään ennustevirheiden estimaatit.

Olkoon $\mathcal{K} : 1, \ldots, N \mapsto 1, \ldots, K$ indeksifunktio, joka kertoo, mihin jakoon havainto j on päätynyt satunnaistamisessa. Merkitään $\hat{\alpha}^{-\mathcal{K}(j)}(\mathbf{x}_j)$ sovitettua mallia, josta on poistettu aineiston k:s osa. Estimaatti ennustetulle virheelle

$$CV = \frac{1}{N} \sum_{j=1}^{N} \lambda(\hat{\alpha}^{-\mathcal{K}(j)}(\mathbf{x}_j)|\omega_{k_j}). \tag{32}$$

Sopivan K:n määrääminen ei ole yksiselitteistä. Suurilla K:n arvoilla harha on pienempi ja ristiinvalidointi estimoi paremmin yleistämisvirhettä $Err_{\mathcal{O}}$. Toisaalta varianssi saattaa olla huomattavasti suurempi. Arvolla $K = N$ päädytään *leave-one-out*-estimaattoriin, jolloin aineistosta poistetaan yksitellen kukin havainto, jolle lasketaan ennustevirhe jäljellä olevista havainnoista. Vaikka arvolla $K = N$ ristiinvalidointi on lähes harhaton, niin sitä ei voi yleisesti suositella suuren varianssin takia (Efron, 1983). Se on myös laskennallisesti vaativin menetelmä. Toisaalta arvolla $K = 2$ tilanne muistuttaa jakoa opetus- ja testiaineistoksi jakosuhteella $50/50$, jolloin harhan suuruus saattaa olla ongelma. Hastie et al. (2009) ehdottavat kompromissiratkaisuna arvoja $K = 5$ ja $K = 10$. Myös ristiinvalidointi voidaan toistaa samalle K useita kertoja (Kim, 2009).

3.3.3 Bootstrap-menetelmät

Bootstrap-ideaa voidaan soveltaa myös ennustevirheen arviointiin. Seuraavaksi esitellään parametrittomia bootstrap-menetelmiä. Myös parametrista bootstrap-menetelmää olisi mahdollista käyttää. Olkoon opetusaineisto $\mathcal{O} = ((\omega_{k_1}, \mathbf{x}_1), \ldots, (\omega_{k_N}, \mathbf{x}_N))$ kokoa N. Bootstrap-menetelmän ideana on tehdä uusia N:n kokoisia otoksia alkuperäisestä opetusaineistosta $\mathcal{O}$ takaisinsijoittaen ja estimoida haluttu tunnusluku. Olkoon kiinnostuksen kohteena aineistosta laskettu tunnusluku T. Olkoon B bootstrap-otosten lukumäärä ja yksittäinen bootstrap-otos $\mathcal{O}^{\star b}$, missä $b = 1, \ldots, B$. Kullekin bootstrap-otokselle lasketaan T, jolloin saadaan bootstrap-replikaatit $T^{\star b}$, $b = 1, \ldots, B$ (Hastie et al, 2009).

Eräs tapa ennustevirheen estimoimiseksi on sovittaa malli erikseen kullakin bootstrap-otoksella, joilla ennustetaan alkuperäistä opetusaineistoa. Tällöin ennustevirheen estimaattori

$$\widehat{Err}_{boot} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{j=1}^{N}\lambda(\hat{\alpha}^{\star b}(\mathbf{x}_j|\omega_{k_j})), \tag{33}$$

missä $\hat{\alpha}^{\star b}(\mathbf{x}_j|\omega_{k_j})$ on bootstrap-otoksen b ennustettu luokka piirteiden $\mathbf{x}_j$ arvolla. Estimaattia $\widehat{Err}_{boot}$ ei voi kuitenkaan yleisesti pitää hyvänä, sillä alkuperäinen opetusaineisto toimii testiaineistona, jolloin kummallakin otoksella on yhteisiä havaintoja (Hastie et al., 2009). Ongelmaa havaintojen päällekkäisyydestä voidaan lähestyä lainaamalla idea ristiinvalidoinnista. Kutakin havaintoa ennustetaan bootstrap-otoksilla, jotka eivät sisällä kyseistä havaintoa. Määritellään ennustevirheelle $leave - one - out$ bootstrap-estimaattori

$$\widehat{Err}^{(1)} = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{C^{-j}}\sum_{b\in C^{-j}}\lambda(\hat{\alpha}^{\star b}(\mathbf{x}_j|\omega_{k_j})), \tag{34}$$

missä C^{-i} on niiden bootstrap-otosten b indeksien joukko, jotka eivät sisällä havaintoa i ja $|C^{-i}|$ sellaisten bootstrap-otosten lukumäärä. Mikäli $|C^{-i}|$ on 0, niin sitä vastaavat termit voidaan jättää pois.

Edellä esitetty $\widehat{Err}^{(1)}$ bootstrap-estimaatti korjaa estimaattorista $\widehat{Err}_{boot}$ aiheutuvaa ylisovittumista. Bootstrap-otoksessa on keskimäärin $0.632 * N$ erillistä havaintoa, sillä

$$\begin{aligned} &Pr(\text{havainto } i \in \text{bootstrap-otos } b) = 1 - (1 - \frac{1}{N})^N \\ &\approx 1 - e^{-1} = 0.632. \end{aligned}$$

Siten sen harha on verrattavissa 2-kertaiseen ristiinvalidointiin, jolloin $\widehat{Err}^{(1)}$ on ylöspäin harhainen todelliselle virheelle. Ratkaisuksi ehdotetaan '.632-estimaattoria' (Efron, 1983), joka voidaan esittää muodossa

$$\widehat{Err}^{(.632)} = 0.368 * \overline{err} + 0.632 * \widehat{Err}^{(1)}. \tag{35}$$

Tämä estimaattori saattaa kuitenkin olla alaspäin harhainen, mikäli luokittelun yhteydessä on ylisovittumista. Efron & Tibshirani (1997) ehdottavat ratkaisuksi '.632+-estimaattoria', joka huomioi ylisovittumisen määrän. Olkoon γ estimaatti virheen todennäköisyydelle tilanteessa, jossa piirteet ja luokat olisivat riippumattomia. Tällöin suhteellinen ylisovittumisen aste

$$\hat{R} = \frac{\widehat{Err}^{(1)} - \overline{err}}{\hat{\gamma} - \overline{err}},$$

jolloin '.632+-estimaattori' voidaan määritellä seuraavasti

$$\widehat{Err}^{(.632+)} = (1 - \hat{w}) \cdot \overline{err} + \hat{w} \cdot \widehat{Err}^{(1)}, \tag{36}$$

jossa

$$\hat{w} = \frac{.632}{1 - .368\hat{R}}.$$

Paino w vaihtelee .632 ja 1 välillä, joten estimaattori $\widehat{Err}^{(.632+)}$ vaihtelee estimaattoreiden $\widehat{Err}^{(.632)}$ ja $\widehat{Err}^{(1)}$ välillä.

4 Bayes-virheen arviointimenetelmien vertailu simulointikokein

Tässä luvussa esitetään Bayes-virheen (3) arviointi käyttäen Monte Carlo -integrointia (28) sekä luvun 3 arviointimenetelmiä, joita ovat ristiinvalidointi (32) ja toistettu ristiinvalidointi eri K:n arvoilla, bootstrap-menetelmät (33) - (36) sekä opetus- ja testiaineistojako eri jakosuhteilla.

Simulointikokeessa piirteet simuloidaan normaalijakaumasta $N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, jossa parametrit $\boldsymbol{\mu}_k$ ja $\boldsymbol{\Sigma}_k$ on estimoitu pohjaeläinaineistosta, joka esitellään seuraavaksi. Mielenkiinnon kohteena on lisäksi, miten menetelmät toimivat pienten aineistojen tapauksessa. Menetelmien vertailu suoritetaan simulointikokein käyttäen kvadraattista luokittelijaa (QDA).

4.1 Aineiston esittely

Tutkimusaineistona käytetään pohjaeläinaineistoa, jota on käytetty tutkimuksissa, joissa on kehitetty pohjaeläinten koneellista tunnistamista (Joutsijoki & Juhola, 2012; Ärje et al., 2013). Aineisto koostuu 50:stä eri taksonomisesta luokasta (ks. Liite B), joista *Habrophlebia sp.*, *Diura sp.* ja *Isoperla sp.* ovat sukuja ja loput lajeja. Pohjaeläimiä on aineistossa yhteensä 6966 kappaletta. Aineiston kuvaamisen ja piirteiden irrottamisen yksityiskohdat on esitelty artikkelissa Ärje et al. (2013). Kunkin pohjaeläimen kuvasta on laskettu yhteensä 64 muuttujaa eli piirrettä (ks. Liite A), jotka on saatu `ImageJ`-ohjelmalla (Rasband, 1997–2010). Osa niistä on harmaasävypiirteitä, jotka kuvaavat harmaasävyjen arvoja kuten keskiarvo, keskihajonta, moodi, minimi, maksimi, massakeskipiste, summa, mediaani, huipukkuus ja vinous. Vastaavat tunnusluvut on laskettu myös RGB-sävyille, jotka kuvaavat puna-, viher- ja sinisävyjen arvoja. Loput ovat geometrisia piirteitä, kuten esimerkiksi pinta-ala, ympärysmitta, leveys ja korkeus. Yksityiskohtaiset tiedot piirteistä on esitetty `ImageJ`-ohjelman manuaalissa (Rasband, 1997). Simulointikokeessa piirteiden määrää on vähennetty.

Taksonomisten luokkien otoskoot ovat välillä 24—633 (Liite B). Huomionarvoista on, että lähes puolet luokista on kooltaan pienempiä kuin piirteiden lukumäärä 64, jolloin kvadraattisen luokitteluanalyysin (QDA) kanssa on ongelmia.

Kuva 4: Kuvia pohjaeläimistä. Ylärivi: *Ameletus inopinatus*, *Arctopsyche ladogensis*, *Asellus aquaticus*. Alarivi: *Baetis niger*, *Baetis rhodani*, *Caenis rivulorum*.

4.2 Synteettisen aineiston Bayes-virhe

Simulointikoetta varten on kiinnitettävä tarkasteltavat luokat sekä piirteet. Olkoon K ennalta tunnettujen luokkien $\omega_1, \ldots, \omega_K$ ja p piirteiden lukumäärä. Prioritodennäköisyydet $P(\omega_k) = \pi_k$ estimoidaan suoraan pohjaeläinaineistosta, ts. havaintoja simuloidaan eri luokista samassa suhteessa kuin pohjaeläinaineistossa. Ehdolliset tiheydet $p(\mathbf{x}|\omega_k) = f_k(\mathbf{x})$ ovat normaalijakaumia $N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, jossa parametrit $\boldsymbol{\mu}_k$ ja $\boldsymbol{\Sigma}_k$ on estimoitu pohjaeläinaineistosta. Otokseen voi tulla myös negatiivisia piirteiden arvoja, jotka eivät välttämättä ole tulkinnallisia. Tästä ei kuitenkaan synny käytännön harmia.

Bayes-virhettä (3) arvioidaan eksaktisti Monte Carlo -integroinnilla, ts. kaavan (27) odotusarvoa arvioidaan otoksesta lasketulla empiirisellä keskiarvolla (28). Aineiston simulointi suoritetaan seuraavanlaisella algoritmilla: Olkoon m simuloitavien havaintojen lukumäärä. Ensin arvotaan havainnolle j luokka ω_{k_j} käyttäen prioritodennäköisyyksiä $P(\omega_1), \ldots, P(\omega_K)$. Tämän jälkeen arvotaan piirteet $\mathbf{x}$ luokan k normaalijakaumasta $N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, $k = 1, \ldots, K$. Ts. arvotaan m kappaletta $(\mathbf{x}, \omega_k)$-pareja, jolloin päätössääntöfunktio muodostuu seuraavasti: Määrätään kullekin havainnolle posterioritodennäköisyydestä $P(\omega_k|\mathbf{x})$ johdettu luokittelupistemäärä $g_k(\mathbf{x})$ kaikille $k = 1, \ldots, K$, jolloin kukin havainto luokitellaan luokkaan, jolle laskettu luokittelupistemäärä on pienin. Kunkin havainnon luokka on ennalta tiedossa, joten väärin luokiteltujen havaintojen lukumäärä saadaan vertaamalla kunkin havainnon luokkaa ennustettuun luokkaan kaavan (28) mukaisesti. Simuloitavien havaintojen lukumäärä m valitaan mahdollisimman suureksi. Väärinluokiteltu havainto noudattaa binomijakaumaa $Bin(1, p_{err})$, jolloin saadaan arvio luokitteluvirheen todennäköisyyden keskivirheelle $\sqrt{p_{err}(1 - p_{err})/m}$, jossa p_{err} on estimaatti Bayes-virheelle P(virhe) (3). Käytännön sovelluksissa Bayes-virhettä ei voida arvioida eksaktisti.

4.3 Arviointimenetelmät

Bayes-virhettä (3) arvioidaan myös luvussa 3 esitetyillä yleisesti käytössä olevilla arviointimenetelmillä. Vertailut tehdään käyttäen kvadraattista luokittelijaa (QDA). Simuloinnissa käytetään erilaisia otoskokoja N riippuen luokkien ja piirteiden lukumäärästä. Lisäksi kiinnitetään simulointien lukumäärä m, ts. simuloidaan useita samankokoisia aineistoja, joiden väärinluokiteltujen osuuksien keskiarvosta saadaan estimaatti Bayes-virheelle. Lisäksi saadaan arvio Bayes-virheen estimaatin keskivirheelle. Kullekin estimaattorille lasketaan myös keskineliösumma MSE, joka saadaan summaamalla estimaattorin varianssi ja harhan neliö. Otoskoko pitää valita niin suureksi, että pystytään estimoimaan päätössääntö opetusaineistosta kaikilla arviointimenetelmillä. Mikäli parametreja ei pystytä estimoimaan jollekin opetusaineistolle, jako opetus- ja testiaineistoksi suoritetaan uudelleen. Kahdelle luokalle ja kahdelle piirteelle käytetään otoskokoja $N = 50, 100, 500, 1000$. Vastaavasti kahdeksan luokan ja kahdeksan piirteen tapauksessa käytetään otoskokoja $N = 500, 1000$ ja viimeisessä simulointikokeessa kaikille 50 luokalle ja 8 piirteelle käytetään otoskokoa $N = 10000$. Kaikissa tapauksissa simuloitavien aineistojen lukumääräksi kiinnitetään $m = 100$, ja samaa simuloitua aineistoa sovelletaan aina kuhunkin arviointimenetelmään. Lisäksi estimoidaan yleistämisvirhe kullekin sadalle simuloidulle ai-

neistolle käyttämällä riippumatonta kokoa $N = 10000$ olevaa testiaineistoa. Tällöin saadaan estimoitua yleistämisvirhe $Err_{\mathcal{O}}$ (30) kullekin opetusaineistolle, ja saatujen estimaattien keskiarvo on estimaatti odotetulle yleistämisvirheelle Err (30).

Kim (2009) sekä Molinaro et al. (2005) valitsivat simulointikokeissaan ristiinvalidoinnin, bootstrap-menetelmien ja opetus- ja testiainejaon toistojen määrät siten, että sovitettavien mallien lukumäärä on yhtä suuri kaikille. Samaa periaatetta sovelletaan myös tässä työssä. Ristiinvalidoinnissa käytetään K:n arvoja 2, 3, 5, 10 ja N. Lisäksi arvolla $K = 5$ ristiinvalidointi toistetaan 10 kertaa ja arvolla $K = 10$ samoin 5 kertaa. Bootstrap-menetelmien kohdalla käytetään estimaattoreita $\widehat{Err}_{boot}$, $\widehat{Err}^{(1)}$, $\widehat{Err}^{(.632)}$ ja $\widehat{Err}^{(.632+)}$. Bootstrap-otosten lukumääräksi kiinnitetään $B = 50$. Opetus- ja testiaineistojaossa jakosuhteena opetusaineistolle käytetään $1/2$, $4/5$ sekä $9/10$ ja jako suoritetaan yhteensä 50 kertaa kullekin aineistolle. Opetusaineiston jokaisessa luokassa täytyy olla vähintään $p + 1$ havaintoa, jotta QDA toimii. Mikäli päätössääntöä ei pystytä estimoimaan jollakin jakokerralla, suoritetaan jako uudelleen.

4.4 Tulokset: 2 luokkaa, 2 piirrettä

Aluksi tarkastellaan yksinkertaista tilannetta, jossa luokkia ja piirteitä on vähän. Valitaan kaksi suurinta luokkaa sekä kaksi 'parhaiten' erottelevaa piirrettä, jotka valitaan `R`-ohjelmointiympäristön `klaR`-paketin `stepclass`-funktiolla. Simulointikokeeseen valittiin kaksi suurinta luokkaa *Taenneb* ja *Isoperla* sekä kaksi 'parhaiten' erottelevaa piirrettä `Mode.green` ja `YM.gray`. Simuloinnit suoritettiin käyttäen pohjaeläinaineistosta estimoituja parametreja, jotka itse asiassa jo esitettiin sivulla 7. Bayes-virheelle saatiin arvio otoskoolla $m = 10^7$: $P(virhe) = 0.0527$ (7e-5) (Taulukko 1).

Bayes-virhe $P(virhe)$ on pienin QDA-luokittelijalla saavutettavissa oleva luokitteluvirheen todennäköisyys tässä piirreavaruudessa. Simulointikokeen perusteella huomataan, että odotettu yleistämisvirhe Err (30) lähestyy Bayes-virhettä opetusaineiston koon kasvaessa (Taulukko 1), mikä oli odotettua. Toisaalta etenkin pienillä otosko'oilla evaluointimenetelmiä olisi järkevää verrata odotettuun yleistämisvirheeseen Err (30), sillä se huomioi opetusaineiston koon.

Otoskoolla $N = 50$ arviointimenetelmät järjestään yliestimoivat Bayes-virhettä. Lähimmäksi pääsee bootstrap-estimaattori $\widehat{Err}_{boot}$, jonka estimoitu harha sekä keskihajonta ovat pienimmät. Parhaan luokittelutuloksen antaa opetusvirhe $\overline{err}$, mutta sen tiedetään antavan liian optimistisia tuloksia. Seuraavaksi pienimmät keskineliösummat ovat bootstrap-estimaattoreilla $\widehat{Err}^{(.632)}$ ja $\widehat{Err}^{(.632+)}$. Näiden välinen ero on odotetusti pieni, sillä ylisovittuminen on vähäistä. Ristiinvalidoinnilla parhaat tulokset saatiin toistetulla ristiinvalidoinnilla K:n arvoilla 5 ja 10 sekä *leave-one-out*-estimaattorilla. Selkeästi huonoiten pärjäsivät ristiinvalidointi K:n arvolla 2 sekä jako opetus- ja testiaineistoksi jakosuhteella $50/50$, mikä johtunee opetusaineiston pienuudesta. Bootstrap-estimaattori $\widehat{Err}^{(1)}$ yliestimoi luokitteluvirhettä, kuten tiedetään ennestään.

Otoskoon kasvaessa myös opetusaineiston koko kasvaa, jolloin harha pienenee. Vastaavasti myös testiaineston koko kasvaa, jolloin keskihajonta pienenee. Erityisesti siis keskineliösummat pienenevät. Otoskoolla $N = 100$ bootstrap-estimaateilla saavutetaan edelleen parhaat tulokset. Erot menetelmien välillä kuitenkin pienenevät otoskoon kasvaessa. Otoskoolla $N = 500$ menetelmien väliset erot pienenevät entisestään. Käytännössä ei ole juurikaan merkitystä, minkä menetelmän tällöin valitsee. Suurimman keskineliösumman antaa opetus- ja testiaineistojako 90/10, mikä johtunee testiaineiston pienuudesta aiheutuvasta varianssista. Otoskoolla $N = 1000$ keskineliösummat ovat jo käytännössä yhtä suuria. Bootstrap-estimaattori $\widehat{Err}_{boot}$ jopa aliestimoi Bayes-virhettä.

4.5 Tulokset: 8 luokkaa, 8 piirrettä

Ensin valittiin kahdeksan suurinta luokkaa: *Aselaqua*, *Baetmuti*, *Diura*, *Isoperla*, *Micrseti*, *Nemoura*, *Protintr* ja *Taenneb* ja kahdeksan 'parasta' piirrettä `Minor`, `Mean.green`, `Mean.gray`, `Mean.blue`, `Major`, `Solidity`, `Median.red` ja `Min.green`. Bayes-virheelle saatiin arvio otoskoolla $m = 10^7$: $P(virhe) = 0.0781$ (8.5e-5) (Taulukko 2).

Vastaavasti nyt bootstrap-estimaattoria $\widehat{Err}_{boot}$ ja opetusvirhettä $\overline{err}$ ei voida pitää hyvinä estimaattoreina luokitteluvirheen todennäköisyydelle. Otoskoolla $N = 500$ paras tulos saadaan *leave-one-out*-ristiinvalidoinnilla. Seuraavaksi parhaat tulokset saadaan bootstrap-estimaattoreilla $\widehat{Err}^{(.632)}$ ja $\widehat{Err}^{(.632+)}$ sekä toistetulla ristiinvalidoinnilla K:n arvolla 10. Selkeästi huonoimmat tulokset saadaan ristiinvalidoinnilla K:n arvolla 2 sekä opetus- ja testiaineistojaolla 50/50. Myös ristiinvalidointi K:n arvolla 3 kärsii opetusaineiston pienuudesta johtuvasta harhasta.

Toistettu ristiinvalidointi K:n arvolla 5 on hyvin verrattavissa opetus- ja testiainejakoon 80/20. Vastaavasti toistettu ristiinvalidointi K:n arvolla 10 vastaa hyvin opetus- ja testiaineistojakoa 90/10. Toistetulla ristiinvalidoinnilla saadaan estimaattorin vaihtelu hieman pienemmäksi.

4.6 Tulokset: 50 luokkaa, 8 piirrettä

Nyt valittuna ovat kaikki 50 luokkaa (Liite B: Taulukko 1) sekä samat 8 piirrettä. Bayes-virheelle saatiin arvio otoskoolla $m = 10^7$: $P(virhe) = 0.1857$ (0.00012) (Taulukko 3). Tässä kokeessa *leave-one-out*-ristiinvalidointi jätettiin pois laskennallisen raskauden takia.

Tulokset ovat hyvin samankaltaiset verrattuna 8 luokan tapaukseen. Paras tulos saavutetaan bootstrap-estimaattoreilla $\widehat{Err}^{(.632)}$ ja $\widehat{Err}^{(.632+)}$. Seuraavaksi parhaat tulokset saadaan toistetulla ristiinvalidoinnilla ja ristiinvalidoinnilla K:n arvolla 10 sekä opetus- ja testiaineistojaolla 90/10. Myös tässä tapauksessa ristiinvalidointi pienillä K:n arvoilla sekä opetus- ja testiaineistojako 50/50 antavat selkeästi huonompia tuloksia.

Arviointimenetelmien eroavaisuuksia voidaan myös hahmotella kuvien 5 ja 6 avulla. Niissä arviointimenetelmien jakaumia verrataan yleistämisvirheen jakaumaan.

Opetusvirhe $\overline{err}$ ja bootstrap-estimaattori $\widehat{Err}_{boot}$ aliestimoivat yleistämisvirhettä. Vastaavasti ristiinvalidointi K:n arvolla 2, opetus- ja testiainestojako 50/50 sekä bootstrap-estimaattori $\widehat{Err}^{(1)}$ yliestimoivat yleistämisvirhettä. Muilta osin eroavaisuudet ovat pieniä.

Taulukko 1: Simulointikokeista eri evaluointimenetelmillä saatuja tunnuslukuja, kun simuloitavia aineistoja 100 ja käytössä 2 luokkaa: *Taenneb* ja *Isoperla* sekä 2 piirrettä: Mode.green ja YM.gray.

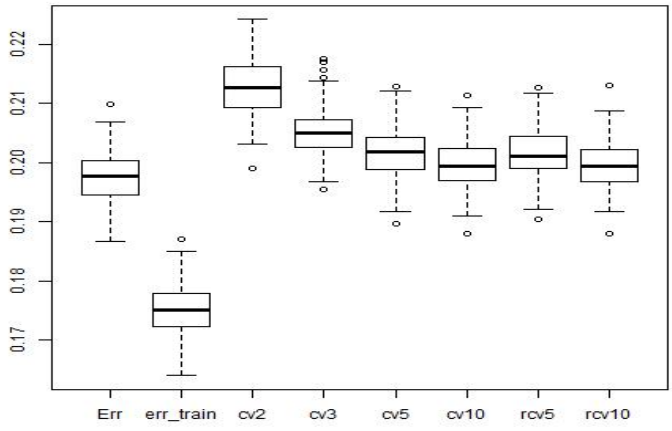
	N=50			N=100		
	$\widehat{Err}$	SD	MSE	$\widehat{Err}$	SD	MSE
$\overline{err}$	.0464	.0307	.00098	.0481	.0226	.00053
CV (K=2)	.0748	.0422	.00227	.0605	.0240	.00064
CV (K=3)	.0664	.0385	.00167	.0592	.0239	.00061
CV (K=5)	.0624	.0359	.00138	.0586	.0235	.00059
CV (K=10)	.0604	.0339	.00121	.0571	.0232	.00056
CV (K=N)	.0602	.0328	.00113	.0554	.0227	.00052
RCV (K=5)	.0629	.0321	.00114	.0574	.0210	.00046
RCV (K=10)	.0610	.0321	.00110	.0561	.0214	.00047
$\widehat{Err}_{boot}$	.0537	.0263	.00069	.0524	.0192	.00037
$\widehat{Err}^{(1)}$	.0706	.0302	.00123	.0604	.0209	.00049
$\widehat{Err}^{(.632)}$	.0617	.0295	.00095	.0559	.0211	.00045
$\widehat{Err}^{(.632+)}$	.0621	.0296	.00096	.0560	.0211	.00045
50/50	.0743	.0304	.00139	.0611	.0212	.00052
80/20	.0626	.0324	.00115	.0570	.0225	.00052
90/10	.0620	.0352	.00133	.0565	.0226	.00052
Err	.0613	.0084		.0567	.0043	
P(virhe)	.0527	7e-5		.0527	7e-5	
	N=500			N=1000		
	$\widehat{Err}$	SD	MSE	$\widehat{Err}$	SD	MSE
$\overline{err}$	.0520	.0105	.00011	.0518	.0065	4e-5
CV (K=2)	.0540	.0105	.00011	.0528	.0062	4e-5
CV (K=3)	.0535	.0107	.00011	.0526	.0066	4e-5
CV (K=5)	.0533	.0109	.00012	.0526	.0064	4e-5
CV (K=10)	.0533	.0105	.00011	.0525	.0063	4e-5
CV (K=N)	.0533	.0107	.00011	.0524	.0065	4e-5
RCV (K=5)	.0535	.0103	.00011	.0527	.0064	4e-5
RCV (K=10)	.0534	.0102	.00011	.0526	.0064	4e-5
$\widehat{Err}_{boot}$	.0528	.0098	.0001	.0522	.0061	4e-5
$\widehat{Err}^{(1)}$	.0542	.0099	.0001	.0529	.0062	4e-5
$\widehat{Err}^{(.632)}$	.0534	.0101	.0001	.0525	.0063	4e-5
$\widehat{Err}^{(.632+)}$	.0534	.0101	.0001	.0525	.0063	4e-5
50/50	.0545	.0100	.0001	.0529	.0063	4e-5
80/20	.0533	.0103	.00011	.0533	.0067	4e-5
90/10	.0534	.0112	.00013	.0525	.0071	5e-5
Err	.0536	.0024		.0532	.0023	
P(virhe)	.0527	7e-5		.0527	7e-5	

Taulukko 2: Simulointikokeista eri evaluointimenetelmillä saatuja tunnuslukuja, kun simuloitavia aineistoja 100 ja käytössä 8 luokkaa sekä 8 piirrettä.

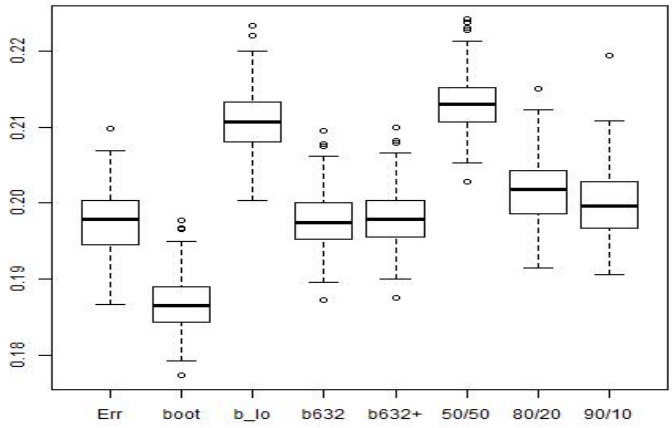
	N=500			N=1000		
	$\widehat{Err}$	SD	MSE	$\widehat{Err}$	SD	MSE
$\overline{err}$	.0572	.0110	.00056	.0667	.0071	.00018
CV (K=2)	.1513	.0197	.00574	.1069	.0101	.00093
CV (K=3)	.1268	.0147	.00258	.0977	.0089	.00046
CV (K=5)	.1163	.0144	.00167	.0938	.0091	.00033
CV (K=10)	.1110	.0139	.00127	.0917	.0089	.00026
CV (K=N)	.1072	.0137	.00103	.0896	.0079	.00020
RCV (K=5)	.1162	.0121	.00160	.0936	.0078	.00030
RCV (K=10)	.1106	.0131	.00123	.0914	.0081	.00024
$\widehat{Err}_{boot}$	.0850	.0090	.00013	.0789	.0065	4e-5
$\widehat{Err}^{(1)}$	.1400	.0117	.00396	.1031	.0077	.00068
$\widehat{Err}^{(.632)}$	.1096	.0107	.00110	.0895	.0072	.00018
$\widehat{Err}^{(.632+)}$	.1116	.0108	.00124	.0899	.0072	.00019
50/50	.1516	.0119	.00554	.1053	.0076	.00080
80/20	.1158	.0125	.00157	.0931	.0083	.00029
90/10	.1116	.0125	.00128	.0909	.0086	.00024
Err	.1054	.0051		.0910	.0036	
P(virhe)	.0781	8.5e-5		.0781	8.5e-5	

Taulukko 3: Simulointikokeista eri evaluointimenetelmillä saatuja tunnuslukuja, kun simuloitavia aineistoja 100 ja käytössä 50 luokkaa sekä 8 piirrettä.

	N=10000		
	$\widehat{Err}$	SD	MSE
$\overline{err}$	.1753	.0042	.00013
CV (K=2)	.2132	.0050	.00078
CV (K=3)	.2055	.0044	.00041
CV (K=5)	.2018	.0043	.00028
CV (K=10)	.1998	.0042	.00022
RCV (K=5)	.2017	.0043	.00027
RCV (K=10)	.1997	.0043	.00021
$\widehat{Err}_{boot}$	.1869	.0040	.00002
$\widehat{Err}^{(1)}$	.2111	.0042	.00066
$\widehat{Err}^{(.632)}$	.1979	.0041	.00017
$\widehat{Err}^{(.632+)}$	.1983	.0041	.00017
50/50	.2133	.0042	.00078
80/20	.2017	.0043	.00027
90/10	.2002	.0048	.00023
Err	.1976	.0044	
P(virhe)	.1857	.00012	



Kuva 5: Simulointikokeen evaluointimenetelmien empiiriset jakaumat, kun simuloitavia aineistoja 100 ja käytössä 50 luokkaa sekä 8 piirrettä (vertaa Taulukko 3). Opetusvirheen ja eri ristiinvalidointimenetelmien jakaumia verrataan yleistämisvirheen $Err_{\mathcal{O}}$ jakaumaan.



Kuva 6: Simulointikokeen evaluointimenetelmien empiiriset jakaumat, kun simuloitavia aineistoja 100 ja käytössä 50 luokkaa sekä 8 piirrettä (vertaa Taulukko 3). Bootstrap-menetelmien ja opetus- ja testiaineistojaon jakaumia verrataan yleistämisvirheen $Err_{\mathcal{O}}$ jakaumaan.

5 Yhteenveto

Otoksen uudelleenkäyttömenetelmiä verrattiin odotettuun yleistämisvirheeseen simulointikokeen avulla, kuten esimerkiksi töissä (Molinaro et al., 2005; Kim, 2009; Borra & Di Ciaccio, 2010). Otoksen uudelleenkäyttömenetelmiä verrattiin myös Bayes-virheeseen, jolla voitiin vertailla paremmin eri otoksen uudelleenkäyttömenetelmiä. Simulointikoe suoritettiin kolmelle eri piirteiden ja luokkien lukumäärien yhdistelmälle. Aluksi luokkia ja piirteitä oli 2, seuraavassa vaiheessa 8 ja lopuksi luokkia oli 50 ja piirteitä 8. Otoskoot olivat välillä $N = 50$ ja $N = 10000$. Luokitteluun käytettiin kvadraattista luokittelijaa (QDA).

Bootstrap-estimaattorit $\widehat{Err}^{(.632)}$ ja $\widehat{Err}^{(.632+)}$ pärjäsivät erittäin hyvin vertailussa kaikissa kolmessa eri simulointitilanteessa. Eroa näiden estimaattoreiden välillä ei kuitenkaan ollut, mikä johtunee vähäisestä ylisovittumisesta. Toistettu ristiinvalidointi $k = 10$ menestyi parhaiten verrattaessa eri ristiinvalidointimenetelmiä sekä opetus- ja testiaineistojakoja. Myös *leave-one-out*-ristiinvalidoinnilla saatiin erinomaisia tuloksia. Sen sijaan etenkin pienillä otoksilla ristiinvalidointi K:n arvoilla 2 ja 3 toimi heikosti. Samoin kävi opetus- ja testiaineistojaon 50/50 tapauksessa. Hastie et al. (2009) suosittelevat kompromissiratkaisuna ristiinvalidointia K:n arvolla 5 tai 10. Molinaro et al. (2005) simulointikokeissa ristiinvalidointi $k = 10$ tuotti hyviä tuloksia lähes kaikissa eri tilanteissa, kun luokittelumenetelminä oli muita menetelmiä kuin QDA, esim. LDA.

Kim (2009) toteaa tutkimuksessaan, että toistetun ristiinvalidoinnin vaihtelun pienenemisestä saatu hyöty on merkittävä. Tässä työssä toistetulla ristiinvalidoinnilla saatiin hieman pienennettyä estimaattorin vaihtelua. Menetelmää voisikin pitää järkevänä, mikäli raskaampi laskenta ei haittaa. Toistettu ristiinvalidointi K:n arvoilla 5 ja 10 tuotti hyvin samankaltaisia tuloksia kuin opetus- ja testiainejako 80/20 ja 90/10. Toistettua opetus- ja testiaineistojakoa voikin pitää siten validina menetelmänä. Etua ristiinvalidointiin nähden ei kuitenkaan ilmennyt simulointikokeessa myöskään tämän tutkimuksen mukaan.

On syytä muistaa, että evaluointimenetelmän paremmuus riippuu otoskoosta. Toisaalta otoskoon kasvaessa menetelmien väliset erot pienenevät. Tässä työssä otoskoot asetettiin niin suuriksi, että luokitteluvirheet saatiin estimoitua järkevästi. Simulointikoetta voisikin laajentaa pudottamalla pois heikoiksi todetut estimaattorit, kuten ristiinvalidointi K:n arvoilla 2 ja 3 sekä opetus- ja testiaineistojako 50/50, jolloin myös otoskokoa voisi pienentää. Luokittelumenetelmänä käytettävä kvadraattinen luokittelija (QDA) asettaa myös rajoitteita: Havaintoja täytyy olla riittävästi joka luokassa, jotta kovarianssimatriisit pystytään estimoimaan. Voisi olla myös mielenkiintoista soveltaa simulointikokeeseen muitakin luokittelumenetelmiä, jolloin myös käytettävien piirteiden määrää voisi lisätä. Lisäksi voisi testata parametrista bootstrap-menetelmää, joka menestyi hyvin Borra Di Ciaccio (2010) simulointikokeissa.

Kiitokset

Kiitokset tutkielmani ohjaajalle FT Salme Kärkkäiselle kaikesta tuesta tutkielman teossa. Hänen positiivinen asenteensa ja kannustustuksensa auttoivat saamaan työn valmiiksi. Kiitokset myös FT Matti Viholalle työn kommentoinnista. Lisäksi kiitokset Suomen ympäristökeskuksen erikoistutkija Kristian Meissnerille pohjaeläinaineistosta sekä emeritusprofessori Antti Penttiselle, FT Johanna Ärjelle ja FT Jouni Helskeelle tutkielmaan liittyvistä neuvoista.

Viitteet

Borra, S. & Di Ciaccio, A. (2010). Measuring the prediction error. A comparison of cross-validation, bootstrap and covariance penalty methods. *Computational Statistics and Data Analysis*, 54 (12): 2976–2989.

Center for Bioinformatics and Computational Biology. URL `http://www.cbcb.umd.edu/~hcorrada/PracticalML/src/classification.R`. Viitattu 17.11.2016.

Duda, R. O., Hart, P. E. & Stork, D. G. (2000). *Pattern Classification*. Wiley, New York, 2nd edition.

Efron, B. (1983). Estimating the error rate of a prediction rule: some improvements on cross-validation. *Journal of the American Statistical Association*, 78 (382): 316–331.

Efron, B. & Tibshirani, R. (1997). Improvements on cross-validation: the 632+ bootstrap: method. *Journal of the American Statistical Association*, 92 (438): 548–560.

Evans, G. (1993). *Practical Numerical Integration*. Wiley, New York.

Friedman, J. H. (1989). Regularized discriminant analysis. *Journal of the American Statistical Association*, 84 (405): 165–175.

Hastie, T., Tibshirani, R. & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer, New York, 2nd edition.

Jain, A. & Zongker, D. (1997). Feature selection: evaluation, application and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19 (2).

Joutsijoki, H. & Juhola, M. (2012). DAGSVM vs. DAGKNN: an experimental case study with benthic macroinvertebrate dataset. *Machine Learning and Data Mining in Pattern Recognition*, 7376: 439–453.

Kim J.-H. (2009). Estimating classification error rate: repeated cross-validation, repeated hold-out and bootstrap. *Computational Statistics and Data Analysis*, 53 (11): 3735–3745.

McLachlan, G. J. (1992). *Discriminant Analysis and Statistical Pattern Recognition*. Wiley & Sons, Inc, New York.

Michie, D., Spiegelhalter D. J. & Taylor C. C. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood Series in Artificial Intelligence, Ellis Horwood.

Molinaro, A. M., Simon R. & Pfeiffer, R. M. (2005). Prediction error estimation: a comparison of resampling methods. *Bioinformatics*, 21: 3301–3307.

R Core Team (2016). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL `http://www.R-project.org/`.

Rasband W. S. (1997). ImageJ manual. URL `http://rsbweb.nih.gov/ij/docs/menus/analyze.html`.

Rasband W. S. (1997–2010). ImageJ, U.S. National Institutes of Health, Bethesda, Maryland, USA. URL `http://rsb.info.nih.gov/ij/`.

Robert, C. & Casella, G. (2009). *Monte Carlo Statistical Methods*. Springer, New York.

Vuori, K.-M., Mitikka, S. & Vuoristo, H. (2009). Pintavesien ekologisen tilan luokittelu. Ympäristöhallinnon ohjeita 3|2009. Helsinki.

Ärje, J., Kärkkäinen, S., Turpeinen, T., & Meissner, K. (2013). Breaking the curse of dimensionality in quadratic discriminant analysis models with a novel variant of a Bayes classifier enhances automated taxa identification of freshwater macroinvertebrates. *Environmetrics*, 24 (4): 248-259.

Liitteet

Liite A: Aineiston piirteet

Piirre	Piirre
Area	StdDev.blue
X	Mode.blue
Y	Min.blue
Perim	Max.blue
BX	XM.blue
BY	YM.blue
Width	IntDen.blue
Height	Median.blue
Major	Skew.blue
Minor	Kurt.blue
Angle	Mean.red
Circ	StdDev.red
Feret	Mode.red
FeretX	Min.red
FeretY	Max.red
FeretAngle	XM.red
MinFeret	YM.red
AR	IntDen.red
Round	Median.red
Solidity	Skew.red
Mean.gray	Kurt.red
StdDev.gray	Mean.green
Mode.gray	StdDev.green
Min.gray	Mode.green
Max.gray	Min.green
XM.gray	Max.green
YM.gray	XM.green
IntDen.gray	YM.green
Median.gray	IntDen.green
Skew.gray	Median.green
Kurt.gray	Skew.green
Mean.blue	Kurt.green

Liite B: Aineiston taksonomiset luokat ja yksilöiden määrät

Taulukko 4: Aineiston taksonomiset luokat ja yksilöiden määrät (n) ko. luokissa.

Luokka	n	Luokka	n
Agapetus	24	*Ameletus inopinatus*	113
Dicranota	27	*Hydraena*	115
Polycentropus	27	*Hydropsyche pellucidulla*	115
Capnia	29	*Ephemera ignita*	116
Gammarus lacustris	30	*Cheumatopsyche lepida*	125
Atherix ibis	31	*Protonemura meyeri*	125
Oulimnius tuberculatus larvae	31	*Baetis rhodani*	130
Lepidostoma hirtum	35	*Hydropsyche saxonica*	161
Ceratopsyche nevae	36	*Limnius volckmari*	167
Leptophlebia	39	*Baetis niger group*	181
Sigas semistriata	39	*Leuctra*	188
Caenis luctuosa	50	*Elmis aenea*	192
Polyflav	52	*Heptagenia dalecarlica*	198
Chimarra marginata	53	*Rhyacophila nubila*	212
Heptagenia sulphurea	53	*Hydropsyche siltalai*	217
Tanypodinae	58	*Ceratopsyche silfvenii*	223
Sericostoma personatum	59	*Ephemara aurivillii*	235
Pisidium	63	*Nemoura*	241
Arctopsyche ladogensis	64	*Diura sp.*	254
Caenis rivulorum	65	*Baetis muticus*	294
Ceratopogonidae	69	*Micrasema setiferum*	295
Ephemera mucronata	69	*Protonemura intricata*	320
Micrasema gelidum	72	*Asellus aquaticus*	339
Callicorixa wollastoni	83	*Isoperla sp.*	535
Habrophlebia sp.	84	*Taeniopteryx nebulosa*	633

Liite C: R-koodi

```
########################

# 2 luokkaa, 2 piirrettä

########################

# Luetaan aineisto
bugs <- read.table("pohjaelaimet.dat", header=TRUE)

myvars <- names(bugs) %in% c("Area.1") # Poistetaan muuttuja Area.1.
bugs2 <- bugs[!myvars]
# Poistetaan tarpeettomat muuttujat id, Label ja Set.
bugs2 <- bugs2[,4:68]

# Valitaan pohjaeläinaineiston 2 isointa luokkaa.
newclasses <- c("Taenneb","Isoperla")

# Etsitään 2 "parasta" piirrettä R-funktiolla stepclass.
# maxvar: valittavien piirteiden määrä
# direction: "forward" eli eteenpäin askeltava algoritmi
# criterion: correctness rate
# fold: 10-kertainen ristiinvalidointi
# method: sovitettava malli QDA

library(klaR)
stepclass(Class ~ ., data=bugs2, maxvar=2, direction="forward",
           criterion="CR", fold=10, method="qda")

# Valitaan 2 "parasta" piirrettä
newvar <- c("Class","Mode.green.","YM.gray.")
newdata <- bugs2[newvar]
newdata <- subset(newdata, Class %in% newclasses)

# Tarvittavat funktiot Bayes-virheen laskemiseksi

########################

# Funktio estimoi aineistosta luokkien prioritodennäköisyydet,
# keskiarvovektorit sekä kovarianssimatriisit.
# Funktion parametreina ovat aineisto ja valitut luokat.
# Aineiston ensimmäisessä sarakkeessa täytyy olla luokat.

calc_stat <- function(data, newclasses){
  new_data <- NULL
```

```
  priors <- NULL
  means <- matrix(0, length(newclasses), ncol(data)-1)
  covars <- list()
  for (i in 1:length(newclasses)){
    new_data <- subset(data, Class==newclasses[i])
    priors[i] <- nrow(new_data)/nrow(data)
    means[i,] <- apply(new_data[,2:ncol(data)], 2, mean)
    covars[[i]] <- cov(new_data[2:ncol(data)])
  }
  return(list(priors=priors, means=means, covars=covars))
}

# Luokittelupistemäärän laskemista varten lasketaan luokan k
# kovarianssimatriisin käänteismatriisi sekä determinantti.
# Funktion parametreina ovat funktion calc_stat.R antamat
# estimaatit aineiston tunnusluvuille sekä luokkien lukumäärä N_k.

invdet_cov <- function(data_sum, N_k){
  covars <- data_sum$covars
  invcovmat <- list()
  detcov <- NULL
  for(k in 1:N_k){
    invcovmat[[k]] <- solve(covars[[k]])
    detcov[k] <- log(det(covars[[k]]))
  }
  return(list(invcovmat=invcovmat, detcov=detcov))
}

# Arvotaan yksilölle luokka ja piirteet, ja lasketaan yksilölle
# kunkin luokan luokittelupistemäärä.
# Pienin luokittelupistemäärä on ennustettu luokka yksilölle.
# Toistetaan sama proseduuri m:lle yksilölle.
# obj: funktion invdet_cov estimaatit
# p_features: piirteiden lukumäärä
# Funktio palauttaa estimaatin Bayes-virheelle.

score <- function(data_sum, obj, N_k, p_features, m){
  invcovmat <- obj$invcovmat
  detcov <- obj$detcov
  means <- data_sum$means
  priors <- data_sum$priors
  score <- NULL
  wrong <- 0
  require(mixtools)
  for(i in 1:m){
    cl <-sample(c(1:N_k), size=1, replace=TRUE, prob=data_sum$priors)
    x <- rmvnorm(1, mu=data_sum$means[cl,],
```

```
                sigma=data_sum$covars[[cl]])
    x <- matrix(x, ncol=1, nrow=p_features)

    for(k in 1:N_k){
      score[k] <- t(x-means[k,])%*%invcovmat[[k]]%*%
      as.matrix(x-means[k,])+detcov[k]-2*log(priors[k])
    }
    pred <- which.min(score)
    if(pred!=cl){
      wrong <- wrong + 1
    }
  }
  Err <- wrong/m
  return(Err)
}

########################

source('D:/R työt/Gradu/Funktiot/calc_stat.R')
source('D:/R työt/Gradu/Funktiot/invdet_cov.R')
source('D:/R työt/Gradu/Funktiot/score.R')

# Bayes-virheen laskeminen

m <- 10000000 # Simuloitavien yksilöiden lukumäärä
N_k <- 2 # Luokkien lukumäärä
p_features <- 2 # Piirteiden lukumäärä

data_sum <- calc_stat(data=newdata, newclasses=newclasses)
invde <- invdet_cov(data_sum=data_sum, N_k=N_k)

seed <- sample(.Machine$integer.max, size=1)
seed # 1572521908
set.seed(seed)

# Estimaatti Bayes-virheelle
p_err <- score(data_sum=data_sum, obj=invde, N_k=N_k,
               p_features=p_features, m=m)

# Estimaatti Bayes-virheen keskivirheelle
sd_p_err <- sqrt(p_err*(1-p_err)/m)

# Otoksen uudelleenkäyttömenetelmät

# Tarvittavat funktiot

########################
```

```
# Arvotaan kullekin yksilölle luokka käyttäen
# estimoituja prioritodennäköisyyksiä.
# Funktion parametreina ovat funktion calc_stat.R
# estimaatit, luokkien lukumäärä N_k ja otoskoko N.

class_sample <- function(data_sum, N_k, N){
  cl <- sample(1:N_k, size=N, replace=TRUE, prob=data_sum$priors)
  return(cl)
}

# Arvotaan kullekin yksilölle piirrevektori.
# Funktion parametreina ovat luokat, funktion data_sum.R
# antamat estimaatit, luokkien lukumäärä N_k, piirteiden
# lukumäärä p_features sekä otoskoko N.
# Funktio palauttaa piirteet matriisina (Nxp-matriisi).

feature_sample <- function(classes, data_sum, N_k, p_features, N){
  require(mixtools)
  features <- matrix(0, nrow=N, ncol=p_features)
  for(k in 1:N_k){
    ind <- which(classes==k)
    features[ind,] <- rmvnorm(length(ind), mu=data_sum$means[k,],
                              sigma=data_sum$covars[[k]])
  }
  return(features)
}

# Ristiinvalidointi
# Funktion parametreina ovat jakosuhde k, piirteet,
# luokat ja otoskoko N.
# jakosuhteeksi käy kokonaisluvut väliltä 2 ja N.

cv_fit <- function(k, features, classes, N){
  require(bootstrap)
  theta.predict <- function(fit, x){predict(fit, x)$class}

  cv.qda <- crossval(x=features, y=classes, theta.fit=qda,
                     theta.predict, ngroup=k)
  err.cv.qda <- 1-sum(cv.qda$cv.fit == classes)/N
  return(err.cv.qda)
}

# Toistettu ristiinvalidointi
# Parametri rep on toistojen lukumäärä.

rcv_fit <- function(k, rep, features, classes, N){
```

```
  require(bootstrap)
  theta.predict <- function(fit, x){predict(fit, x)$class}

  err.cv.qda <- NULL
  for(i in 1:rep){
    cv.qda <- crossval(x=features, y=classes, theta.fit=qda,
                       theta.predict, ngroup=k)
    err.cv.qda[i] <- 1-sum(cv.qda$cv.fit == classes)/N
  }
  err.rcv.qda <- mean(err.cv.qda)
  return(err.rcv.qda)
}

# Toistettu opetus- ja testiainejako
# Jakosuhde k annetaan murtolukuna (esimerkiksi k=1/2).
# rep: opetus- ja testiaineistojaon toistojen lukumäärä.
# Parametrina annetaan myös data, jonka ensimmäisessä sarakkeessa
# täytyy olla luokat.
# Jako opetus- ja testiaineistoksi suoritetaan uudelleen, mikäli
# jossakin opetusaineiston luokassa on alle p+1 yksilöä.

rhold_out <- function(data, N, k, rep){
  Err <- NULL
  for(i in 1:rep){
    repeat {
      train_rows <- sample(1:N,(k)*N)
      data_train <- data[train_rows,]
      data_test <- data[-train_rows,]
      if(all(table(data_train$classes)>ncol(data)-1))
        break
    }
    QDA <- qda(data_train[,2:ncol(data)], data_train$classes)
    QDA.pred <- predict(QDA, data_test[,2:ncol(data)])$class
    Err[i] <- 1-sum(QDA.pred==data_test$classes)/nrow(data_test)
  }
  error <- mean(Err)
  return(error=error)
}

# Testivirheen/yleistämisvirheen laskeminen
# Opetusaineiston koko on N_train ja
# testiaineiston koko N_test.

test_err <- function(data_sum, N_train, N_test, N_k, p_features){
  err_o <- NULL
  N <- N_train + N_test
  for(i in 1:N_sim){
```

```
    set.seed(i)
    classes <- class_sample(data_sum=data_sum, N_k=N_k, N=N)
    features <- feature_sample(classes=classes, data_sum=data_sum,
                               N_k=N_k, p_features=p_features, N=N)
    data <- data.frame(cbind(classes,features))
    train_rows <- 1:N_train
    data_train <- data[train_rows,]
    data_test <- data[-train_rows,]
    QDA <- qda(data_train[,2:ncol(data)], data_train$classes)
    QDA.pred <- predict(QDA, data_test[,2:ncol(data)])$class
    err_o[i] <- 1-sum(QDA.pred==data_test$classes)/nrow(data_test)
  }
  return(err_o=err_o)
}

########################

source('D:/R työt/Gradu/Funktiot/calc_stat.R')
source('D:/R työt/Gradu/Funktiot/class_sample.R')
source('D:/R työt/Gradu/Funktiot/feature_sample.R')
source('D:/R työt/Gradu/Funktiot/cv_fit.R')
source('D:/R työt/Gradu/Funktiot/rcv_fit.R')
source('D:/R työt/Gradu/Funktiot/rhold_out.R')
source('D:/R työt/Gradu/Funktiot/test_err.R')

data_sum <- calc_stat(data=newdata, newclasses=newclasses)

N <- 50 # Simuloitavien havaintojen lukumäärä
# Myös N=100, N=500 ja N=1000

N_sim <- 100 # Simuloitavien aineistojen lukumäärä
N_k <- 2 # Luokkien lukumäärä
p_features <- 2 # Piirteiden lukumäärä

# Alustetaan muuttujat
err_train <- NULL; err_cv2 <- NULL; err_cv3 <- NULL;
err_cv5 <- NULL; err_cv10 <- NULL; err_cvlo <- NULL;
err_rcv5 <- NULL; err_rcv10 <- NULL; err_boot <- NULL;
err_loo_boot <- NULL; err_632 <- NULL; err_632plus <- NULL;
err_ho50 <- NULL; err_ho80 <- NULL; err_ho90 <- NULL

# Funktio bootstrap-estimaattorien ja opetusvirheen laskemiseen
library(sortinghat)
qda_wrapper <- function(object, newdata) {
  predict(object, newdata)$class
}
```

```
for(i in 1:N_sim){
  set.seed(i)
  classes <- class_sample(data_sum=data_sum, N_k=N_k, N=N)
  features <- feature_sample(classes=classes, data_sum=data_sum,
                             N_k=N_k, p_features=p_features, N=N)

  sim_data <- data.frame(cbind(classes,features))

  set.seed(i); err_train[i] <- errorest(x=features, y=classes,
                                        estimator="apparent",
                                        train=MASS:::qda,
                                        classify=qda_wrapper)

  set.seed(i); err_cv2[i] <- cv_fit(k=2, features=features,
                                    classes=classes, N=N)
  set.seed(i); err_cv3[i] <- cv_fit(k=3, features=features,
                                    classes=classes, N=N)
  set.seed(i); err_cv5[i] <- cv_fit(k=5, features=features,
                                    classes=classes, N=N)
  set.seed(i); err_cv10[i] <- cv_fit(k=10, features=features,
                                     classes=classes, N=N)
  set.seed(i); err_cvlo[i] <- cv_fit(k=N, features=features,
                                     classes=classes, N=N)

  set.seed(i); err_rcv5[i] <- rcv_fit(k=5, rep=10, features=features,
                                      classes=classes, N=N)
  set.seed(i); err_rcv10[i] <- rcv_fit(k=10, rep=5, classes=classes,
                                       features=features, N=N)

  set.seed(i); err_boot[i] <- errorest(x=features, y=classes,
                                       estimator="boot",
                                       train=MASS:::qda,
                                       classify=qda_wrapper,
                                       num_bootstraps=50)

  set.seed(i); err_loo_boot[i] <- errorest(x=features, y=classes,
                                           estimator="loo-boot",
                                           train=MASS:::qda,
                                           classify=qda_wrapper,
                                           num_bootstraps=50)

  set.seed(i); err_632[i] <- errorest(x=features, y=classes,
                                      estimator="632",
                                      train=MASS:::qda,
                                      classify=qda_wrapper,
                                      num_bootstraps=50)
```

```
  set.seed(i); err_632plus[i] <- errorest(x=features, y=classes,
                                         estimator="632+",
                                         train=MASS:::qda,
                                         classify=qda_wrapper,
                                         num_bootstraps=50)

  set.seed(i); err_ho50[i] <- rhold_out(data=sim_data, N=N,
                                        k=1/2, rep=50)
  set.seed(i); err_ho80[i] <- rhold_out(data=sim_data, N=N,
                                        k=4/5, rep=50)
  set.seed(i); err_ho90[i] <- rhold_out(data=sim_data, N=N,
                                        k=9/10, rep=50)
}

# Aiemmin laskettu Bayes-virhe 2 luokalle ja 2 piirteelle
p_error <- 0.0527183

# Estimaatit luokitteluvirheille, keskihajonnoille sekä
# keskineliösummille eri otoksen uudelleenkäyttömenetelmillä
round(mean(err_train), 4); round(sd(err_train), 4)
round(sd(err_train)^2 + (mean(err_train) - p_error)^2, 5)

round(mean(err_cv2), 4); round(sd(err_cv2), 4)
round(sd(err_cv2)^2 + (mean(err_cv2) - p_error)^2, 5)

round(mean(err_cv3), 4); round(sd(err_cv3), 4)
round(sd(err_cv3)^2 + (mean(err_cv3) - p_error)^2, 5)

round(mean(err_cv5), 4); round(sd(err_cv5), 4)
round(sd(err_cv5)^2 + (mean(err_cv5) - p_error)^2, 5)

round(mean(err_cv10), 4); round(sd(err_cv10), 4)
round(sd(err_cv10)^2 + (mean(err_cv10) - p_error)^2, 5)

round(mean(err_cvlo), 4); round(sd(err_cvlo), 4)
round(sd(err_cvlo)^2 + (mean(err_cvlo) - p_error)^2, 5)

round(mean(err_rcv5), 4); round(sd(err_rcv5), 4)
round(sd(err_rcv5)^2 + (mean(err_rcv5) - p_error)^2, 5)

round(mean(err_rcv10), 4); round(sd(err_rcv10), 4)
round(sd(err_rcv10)^2 + (mean(err_rcv10) - p_error)^2, 5)

round(mean(err_boot), 4); round(sd(err_boot), 4)
round(sd(err_boot)^2 + (mean(err_boot) - p_error)^2, 5)

round(mean(err_loo_boot), 4); round(sd(err_loo_boot), 4)
```

```
round(sd(err_loo_boot)^2 + (mean(err_loo_boot) - p_error)^2, 5)

round(mean(err_632), 4); round(sd(err_632), 4)
round(sd(err_632)^2 + (mean(err_632) - p_error)^2, 5)

round(mean(err_632plus), 4); round(sd(err_632plus), 4)
round(sd(err_632plus)^2 + (mean(err_632plus) - p_error)^2, 5)

round(mean(err_ho50), 4); round(sd(err_ho50), 4)
round(sd(err_ho50)^2 + (mean(err_ho50) - p_error)^2, 5)

round(mean(err_ho80), 4); round(sd(err_ho80), 4)
round(sd(err_ho80)^2 + (mean(err_ho80) - p_error)^2, 5)

round(mean(err_ho90), 4); round(sd(err_ho90), 4)
round(sd(err_ho90)^2 + (mean(err_ho90) - p_error)^2, 5)

# Estimoidaan yleistämisvirhe 100 eri opetusaineistolle
# opetusaineiston ko'oilla 50, 100, 500 ja 1000.
err_o_50 <- test_err(data_sum=data_sum, N_train=50,
                     N_test=10000, N_k=2, p_features=2)
err_o_100 <- test_err(data_sum=data_sum, N_train=100,
                      N_test=10000, N_k=2, p_features=2)
err_o_500 <- test_err(data_sum=data_sum, N_train=500,
                      N_test=10000, N_k=2, p_features=2)
err_o_1000 <- test_err(data_sum=data_sum, N_train=1000,
                       N_test=10000, N_k=2, p_features=2)

# Estimaatit yleistämisvirheille ja niiden keskihajonnoille
mean(err_o_50); mean(err_o_100); mean(err_o_500); mean(err_o_1000)
sd(err_o_50); sd(err_o_100); sd(err_o_500); sd(err_o_1000)

########################

# 8 luokkaa, 8 piirrettä

########################

# Luetaan aineisto
bugs <- read.table("pohjikset50.dat", header=TRUE)

myvars <- names(bugs) %in% c("Area.1") # Poistetaan muuttuja Area.1.
bugs2 <- bugs[!myvars]
# Poistetaan tarpeettomat muuttujat id, Label ja Set.
bugs2 <- bugs2[,4:68]

# Valitaan 8 isointa luokkaa.
```

```
clas <- levels(bugs2$Class)
l_cl <- table(bugs2$Class)>240
newclasses <- clas[l_cl]

# Etsitään 8 "parasta" piirrettä R-funktiolla stepclass.
stepclass(Class ~ ., data = bugs2, maxvar=8, direction = "forward",
          criterion = "CR", fold = 10, method = "qda")

newvar <- c("Class","Minor","Mean.green.","Mean.gray.","Mean.blue.",
            "Major","Solidity","Median.red.","Min.green.")
newdata <- bugs2[newvar]
newdata <- subset(newdata, Class %in% newclasses)

# Tarvittavat funktiot Bayes-virheen laskemiseksi

source('D:/R työt/Gradu/Funktiot/calc_stat.R')
source('D:/R työt/Gradu/Funktiot/invdet_cov.R')
source('D:/R työt/Gradu/Funktiot/score.R')

# Bayes-virheen laskeminen

m <- 10000000 # Simuloitavien yksilöiden lukumäärä
N_k <- 8 # Luokkien lukumäärä
p_features <- 8 # Piirteiden lukumäärä

data_sum <- calc_stat(data=newdata, newclasses=newclasses)
invde <- invdet_cov(data_sum=data_sum, N_k=N_k)

seed <- sample(.Machine$integer.max, size=1)
seed # 495870689
set.seed(seed)

# Estimaatti Bayes-virheelle
p_err <- score(data_sum=data_sum, obj=invde, N_k=N_k,
               p_features=p_features, m=m)

# Estimaatti Bayes-virheen keskivirheelle
sd_p_err <- sqrt(p_err*(1-p_err)/m)

# Otoksen uudelleenkäyttömenetelmät

data_sum <- calc_stat(data=newdata, newclasses=newclasses)

N <- 500 # Simuloitavien havaintojen lukumäärä
# Myös N=1000

N_sim <- 100 # Simuloitavien aineistojen lukumäärä
```

```
N_k <- 8 # Luokkien lukumäärä
p_features <- 8 # Piirteiden lukumäärä

# Aiemmin laskettu Bayes-virhe 8 luokalle ja 8 piirteelle
p_error <- 0.0781255

# Estimoidaan yleistämisvirhe 100 eri opetusaineistolle
# opetusaineiston ko'oilla 500 ja 1000.
err_o_500 <- test_err(data_sum=data_sum, N_train=500,
                      N_test=10000, N_k=8, p_features=8)
err_o_1000 <- test_err(data_sum=data_sum, N_train=1000,
                       N_test=10000, N_k=8, p_features=8)

# Estimaatit yleistämisvirheille ja niiden keskihajonnoille
mean(err_o_500); mean(err_o_1000)
sd(err_o_500); sd(err_o_1000)

########################

# 50 luokkaa, 8 piirrettä

########################

# Luetaan aineisto
bugs <- read.table("pohjikset50.dat", header=TRUE)

myvars <- names(bugs) %in% c("Area.1") # Poistetaan muuttuja Area.1.
bugs2 <- bugs[!myvars]
# Poistetaan tarpeettomat muuttujat id, Label ja Set.
bugs2 <- bugs2[,4:68]

# Valitaan kaikki luokat.
newclasses <- levels(bugs2$Class)

# Käytetään samoja 8 piirrettä.
newvar <- c("Class","Minor","Mean.green.","Mean.gray.","Mean.blue.",
            "Major","Solidity","Median.red.","Min.green.")
newdata <- bugs2[newvar]
newdata <- subset(newdata, Class %in% newclasses)

# Tarvittavat funktiot Bayes-virheen laskemiseksi

source('D:/R työt/Gradu/Funktiot/calc_stat.R')
source('D:/R työt/Gradu/Funktiot/invdet_cov.R')
source('D:/R työt/Gradu/Funktiot/score.R')

# Bayes-virheen laskeminen
```

```
m <- 10000000 # Simuloitavien yksilöiden lukumäärä
N_k <- 50 # Luokkien lukumäärä
p_features <- 8 # Piirteiden lukumäärä

data_sum <- calc_stat(data=newdata, newclasses=newclasses)
invde <- invdet_cov(data_sum=data_sum, N_k=N_k)

seed <- sample(.Machine$integer.max, size=1)
seed # 1879981471
set.seed(seed)

# Estimaatti Bayes-virheelle
p_err <- score(data_sum=data_sum, obj=invde, N_k=N_k,
               p_features=p_features, m=m)

# Estimaatti Bayes-virheen keskivirheelle
sd_p_err <- sqrt(p_err*(1-p_err)/m)

# Otoksen uudelleenkäyttömenetelmät

data_sum <- calc_stat(data=newdata, newclasses=newclasses)

N <- 10000 # Simuloitavien havaintojen lukumäärä
N_sim <- 100 # Simuloitavien aineistojen lukumäärä
N_k <- 50 # Luokkien lukumäärä
p_features <- 8 # Piirteiden lukumäärä

# Aiemmin laskettu Bayes-virhe 50 luokalle ja 8 piirteelle
p_error <- 0.1857347

# err_cvlo jätetään pois laskennallisen raskauden takia

# Estimoidaan yleistämisvirhe 100 eri opetusaineistolle
# opetusaineiston koolla 10000.
err_o_10000 <- test_err(data_sum=data_sum, N_train=10000,
                        N_test=10000, N_k=50, p_features=8)

# Estimaatti yleistämisvirheelle ja sen keskihajonnalle
mean(err_o_10000)
sd(err_o_10000)

# QDA:ta ei pystynyt sovittamaan muutamassa tilanteessa.
# Tällöin tehtiin samalle aineistolle uudet bootstrap-otokset tai
# ristiinvalidoinnin tapauksessa uusi ositus aineistolle.

########################
```

```
err_50_8 <- read.table("err_50_8.csv", header=TRUE, sep = "/")
# Err_boot
# QDA ei toiminut indekseillä 66 ja 70.
# Suoritetaan uudet bootstrap-otokset kyseisille aineistoille.

set.seed(66)
classes <- class_sample(data_sum=data_sum, N_k=N_k, N=N)
features <- feature_sample(classes=classes, data_sum=data_sum,
                           N_k=N_k, p_features=p_features, N=N)
seed <- sample(.Machine$integer.max, size=1)
seed # 955964697
set.seed(seed); err_boot[66] <- errorest(x=features, y=classes,
                                         estimator="boot",
                                         train=MASS:::qda,
                                         classify=qda_wrapper,
                                         num_bootstraps=50)

set.seed(70)
classes <- class_sample(data_sum=data_sum, N_k=N_k, N=N)
features <- feature_sample(classes=classes, data_sum=data_sum,
                           N_k=N_k, p_features=p_features, N=N)
seed <- sample(.Machine$integer.max, size=1)
seed # 1827743933
set.seed(seed); err_boot[70] <- errorest(x=features, y=classes,
                                         estimator="boot",
                                         train=MASS:::qda,
                                         classify=qda_wrapper,
                                         num_bootstraps=50)

# Err_loo_boot
# QDA ei toiminut indeksillä 70.
# Suoritetaan uudet bootstrap-otokset kyseiselle aineistolle.

set.seed(70)
classes <- class_sample(data_sum=data_sum, N_k=N_k, N=N)
features <- feature_sample(classes=classes, data_sum=data_sum,
                           N_k=N_k, p_features=p_features, N=N)
seed <- sample(.Machine$integer.max, size=1)
seed # 597452550
set.seed(seed); err_loo_boot[70] <- errorest(x=features, y=classes,
                                             estimator="loo-boot",
                                             train=MASS:::qda,
                                             classify=qda_wrapper,
                                             num_bootstraps=50)

# err_cv2
```

```
# QDA ei toiminut indekseillä 1, 4, 31 ja 43.
# Suoritetaan uudet bootstrap-otokset kyseisille aineistoille.

set.seed(1)
classes <- class_sample(data_sum=data_sum, N_k=N_k, N=N)
features <- feature_sample(classes=classes, data_sum=data_sum,
                           N_k=N_k, p_features=p_features, N=N)
seed <- sample(.Machine$integer.max, size=1)
seed # 463155743
set.seed(seed); err_cv2[1] <- cv_fit(k=2, features=features,
                                     classes=classes, N=N)

set.seed(4)
classes <- class_sample(data_sum=data_sum, N_k=N_k, N=N)
features <- feature_sample(classes=classes, data_sum=data_sum,
                           N_k=N_k, p_features=p_features, N=N)
seed <- sample(.Machine$integer.max, size=1)
seed # 1547617708
set.seed(seed); err_cv2[4] <- cv_fit(k=2, features=features,
                                     classes=classes, N=N)

set.seed(31)
classes <- class_sample(data_sum=data_sum, N_k=N_k, N=N)
features <- feature_sample(classes=classes, data_sum=data_sum,
                           N_k=N_k, p_features=p_features, N=N)
seed <- sample(.Machine$integer.max, size=1)
seed # 1819309678
set.seed(seed); err_cv2[31] <- cv_fit(k=2, features=features,
                                      classes=classes, N=N)

set.seed(43)
classes <- class_sample(data_sum=data_sum, N_k=N_k, N=N)
features <- feature_sample(classes=classes, data_sum=data_sum,
                           N_k=N_k, p_features=p_features, N=N)
seed <- sample(.Machine$integer.max, size=1)
seed # 1757405517
set.seed(seed); err_cv2[43] <- cv_fit(k=2, features=features,
                                      classes=classes, N=N)

########################

# Kootaan tulokset matriisiksi.
err_50_8 <- cbind(err_train, err_cv2, err_cv3, err_cv5,
                  err_cv10, err_rcv5, err_rcv10, err_boot,
                  err_loo_boot, err_632, err_632plus,
                  err_ho50, err_ho80, err_ho90)
```

```
# Estimaattorien jakaumat
boxplot(err_o_10000, err_50_8[,1], err_50_8[,2], err_50_8[,3],
        err_50_8[,4], err_50_8[,5], err_50_8[,6], err_50_8[,7],
        names=c("Err","err_train","cv2","cv3",
                "cv5","cv10","rcv5","rcv10"))

boxplot(err_o_10000, err_50_8[,8], err_50_8[,9], err_50_8[,10],
        err_50_8[,11], err_50_8[,12], err_50_8[,13], err_50_8[,14],
        names=c("Err","boot","b_lo","b632","b632+",
                "50/50","80/20","90/10"))

#########################

# Päätössäännöt QDA:lla ja LDA:lla

# Luetaan aineisto
bugs <- read.table("pohjaelaimet.dat", header=TRUE)

myvars <- names(bugs) %in% c("Area.1") # Poistetaan muuttuja Area.1.
bugs2 <- bugs[!myvars]
# Poistetaan tarpeettomat muuttujat id, Label ja Set.
bugs2 <- bugs2[,4:68]

# Valitaan pohjaeläinaineiston 2 isointa luokkaa.
newclasses <- c("Taenneb","Isoperla")

# Valitaan 2 "parasta" piirrettä
newvar <- c("Class","Mode.green.","YM.gray.")
newdata <- bugs2[newvar]
newdata <- subset(newdata, Class %in% newclasses)

source('D:/R työt/Gradu/Funktiot/calc_stat.R')
source('D:/R työt/Gradu/Funktiot/class_sample.R')
source('D:/R työt/Gradu/Funktiot/feature_sample.R')

N <- 100 # Simuloitavien yksilöiden lukumäärä
N_k <- 2 # Luokkien lukumäärä
p_features <- 2 # Piirteiden lukumäärä

data_sum <- calc_stat(data=newdata, newclasses=newclasses)

seed <- sample(.Machine$integer.max, size=1)
seed # 1048056859
set.seed(seed)

classes <- class_sample(data_sum=data_sum, N_k=N_k, N=N)
features <- feature_sample(classes=classes, data_sum=data_sum,
```

```
                          N_k=N_k, p_features=p_features, N=N)

# Nimetään mallin parametrit uudelleen
mu1 <- c(data_sum$means[1,1], data_sum$means[1,2])
mu2 <- c(data_sum$means[2,1], data_sum$means[2,2])
cov1 <- data_sum$covars[[1]]
cov2 <- data_sum$covars[[2]]
cov_pooled <- (cov1+cov2)/2

# Nimetään myös muuttujat uudelleen
Y <- classes
X1 <- features[,1]
X2 <- features[,2]
fit.lda <- lda(Y~X1+X2)
fit.qda <- qda(Y~X1+X2)

# Päätössäänöt piirretty sivun http://www.cbcb.umd.edu/
# ~hcorrada/PracticalML/src/classification.R merkinnöillä
# Luodaan 500 x 500 hila
GS <- 500
XLIM <- range(X1)
tmpx <- seq(XLIM[1], XLIM[2], len=GS)
YLIM <- range(X2)
tmpy <- seq(YLIM[1], YLIM[2], len=GS)
newx <- expand.grid(tmpx, tmpy)

Ghat.lda <- as.numeric(predict(fit.lda, newdata=data.frame(
                        X1=newx[,1],X2=newx[,2]))$class)
Ghat.qda <- as.numeric(predict(fit.qda, newdata=data.frame(
                        X1=newx[,1],X2=newx[,2]))$class)

# Piirretään gaussiset jakaumat QDA:lle
par(mfrow=c(1,2), no.readonly = TRUE)
plot(X1, X2, col=classes, pch=classes, cex=1.25,
     xlab="x1",ylab="x2")
ellipse(mu1, cov1, npoints = 200, newplot = FALSE)
ellipse(mu2, cov2, npoints = 200, newplot = FALSE)

# Piirretään päätössääntö QDA:lle
plot(X1, X2, col=classes, pch=classes, cex=1.25,
     xlab="x1",ylab="x2")
contour(tmpx, tmpy, matrix(Ghat.qda, GS, GS), levels=c(1,2),
        add=TRUE, drawlabels=FALSE)

# Piirretään gaussiset jakaumat LDA:lle
par(mfrow=c(1,2), no.readonly = TRUE)
plot(X1, X2, col=classes, pch=classes, cex=1.25,
```

```
     xlab="x1",ylab="x2")
ellipse(mu1, cov_pooled, npoints = 200, newplot = FALSE)
ellipse(mu2, cov_pooled, npoints = 200, newplot = FALSE)

# Piirretään päätössääntö LDA:lle
plot(X1, X2, col=classes, pch=classes, cex=1.25,
     xlab="x1",ylab="x2")
contour(tmpx, tmpy, matrix(Ghat.lda, GS, GS), levels=c(1,2),
        add=TRUE, drawlabels=FALSE)

# Chernoffin yläraja

chernoff <- function(beta, mu1, mu2, cov1, cov2, priors){
  k_beta <- NULL
  e_kbeta <- NULL
  for(i in 1:length(beta)){
    k_beta[i] <- (beta[i]*(1-beta[i])/2)*t(mu2-mu1)%*%
      solve(beta[i]*cov1+(1-beta[i])*cov2)%*%(mu2-mu1)+
      0.5*log(det(beta[i]*cov1+(1-beta[i])*cov2)/
                ((det(cov1)^beta[i])*det(cov2)^(1-beta[i])))
    e_kbeta[i] <- exp(-k_beta[i])
  }
  min_ekbeta <- min(e_kbeta)
  min_beta <- beta[which.min(e_kbeta)]
  cher_min <- (priors[1]^min_beta)*(priors[2]^(1-min_beta))*
    min(e_kbeta)
  return(list(beta=beta, e_kbeta=e_kbeta, min_ekbeta=min_ekbeta,
              min_beta=min_beta, cher_min=cher_min))
}

cher <- chernoff(beta=seq(0,1, by=0.001), mu1=mu1, mu2=mu2,
                 cov1=cov1, cov2=cov2, priors=data_sum$priors)
bhat <- chernoff(beta=0.5, mu1=mu1, mu2=mu2, cov1=cov1, cov2=cov2,
                 priors=data_sum$priors)

plot(cher$beta, cher$e_kbeta, ylim=c(0,1), ylab="exp(-k(beta))")
segments(x0 = cher$min_beta, y0 = -0.04, x1 = cher$min_beta,
         y1 = cher$min_ekbeta, col = "red", lwd=2)
segments(x0 = -0.04, y0 = cher$min_ekbeta, x1 = cher$min_beta,
         y1 = cher$min_ekbeta, col = "red", lwd=2)
segments(x0 = 0.5, y0 = -0.04, x1 = 0.5, y1 = bhat$min_ekbeta,
         col = "blue", lwd=2, lty=2)
segments(x0 = -0.04, y0 = 0.2414074, x1 = 0.5, y1 = bhat$min_ekbeta,
         col = "blue", lwd=2, lty=2)
legend(0.35, 0.75, c("Chernoff", "Bhattacharya"),
       col = c("red", "blue"), lty = c(1, 2))
```