Vahvistetut päätöspuumallit

Pro Gradu
Juho Luukkonen
Matemaattisten tieteiden tutkimusyksikkö
Oulun yliopisto
Kevät 2020

Sisällys

Algoritmit

1 Johdanto

Tässä työssä käsitellään ohjatun oppimisen menetelmiä. Lähtökohta on seuraavanlainen: Käytettävissä on otos $\{(Y_1, \mathbf{X}_1), (Y_2, \mathbf{X}_2), \ldots, (Y_N, \mathbf{X}_N)\}$ havaintoja jostakin $(Y, \mathbf{X})$-yhteisjakaumasta, missä $\mathbf{X}_i = (X_i^{(1)}, X_i^{(2)}, \ldots, X_i^{(N)})$, $i = 1, \ldots, N$. Tavoitteena on löytää estimaatti $\hat{f}(\mathbf{X})$ funktiolle $f^* \colon \mathbf{X} \mapsto Y$, joka minimoi tappiofunktion L odotusarvon yli $(Y, \mathbf{X})$-yhteisjakauman:

$$f^* = \arg\min_f E_{Y,\mathbf{X}} L(Y, f(\mathbf{X})) = \arg\min_f E_{\mathbf{X}}[E_Y(L(Y, f(\mathbf{X})))|\mathbf{X}]. \quad (1)$$

Kappaleessa 2 esitetään päätöspuiden eli luokittelu- ja regressiopuiden yleistä teoriaa. Kappale 3 alkaa AdaBoostin, varhaisen vahvistamisalgoritmin, esittelyllä, ja tämän jälkeen jatkuu yleisemmin vaiheittaisten additiivisten mallien ja virhekriteerien tarkastelulla. Alikappaleessa 3.8 esitellään hieman toisenlainen tulkinta AdaBoostista ja vahvistamisesta ylipäätään. Tämän jälkeen yhdistetään kappaleiden 2 ja 3 sisältö ja esitellään vahvistettujen puiden teoriaa, ja tästä tärkeänä erikoistapauksena gradienttivahvistaminen puilla. Seuraavaksi esitellään regularisaatiomenetelmiä, eli menetelmiä hillitä mallin tarpeetonta kompleksisuutta, laskennallista "hintaa", ja ylisovittumista. Kappaleen 3 lopussa tarkastellaan vielä vahvistettujen puumallien tulkintaa. Kappaleessa 4 esitellään sovitetun gradienttivahvistusmallin tuloksia simuloidulla havaintoaineistolla.

Kappaleiden 2 ja 3 pääasiallisena lähteenä on Hastien, Friedmanin ja Tibshiranin kirja [9], lukuunottamatta alikappaletta 3.8 joka perustuu Breimanin artikkeliin [2].

Tämän työn pääteemat ovat päätöspuumallit, vahvistaminen, ja päätöspuumallien vahvistaminen. Perusteos, jossa ensimmäisenä esitellään monet tässä työssä esitellyt päätöspuihin liittyvät käsitteet nykyisessä muodossaan on [1]. Niitä on käsitelty myöhemmin lukuisissa teoksissa, mm. [11] luokittelupuiden osalta. Vahvistamisesta, sen tulkinnasta ja paikasta laajemmassa teoreettisessa viitekehyksessä, ja siitä miksi se toimii, on käyty keskustelua ja esitetty erilaisia näkemyksiä, mm. [12], [2], [3], [4], [6]. Friedman on kehittänyt vahvistamismenetelmiä edelleen *gradienttivahvistamisen* [7] ja *stokastisen gradienttivahvistamisen* [5] muodossa.

2 Puupohjaiset menetelmät

Puupohjaiset menetelmät jakavat piirreavaruuden joukkoon suorakaiteita, ja sovittavat yksinkertaisen mallin (kuten vakion) jokaiseen näistä. Konsepti on

yksinkertainen mutta tehokas. Kuvaillaan aluksi puupohjaisessa regressiossa ja luokittelussa suosittu menetelmä.

Tarkastellaan regressio-ongelmaa jatkuvalla vasteella Y ja muuttujilla X_1, X_2, jotka molemmat saavat arvoja yksikköintervallilla.
Jaamme ensiksi avaruuden kahteen alueeseen, ja mallinnamme vastetta Y kummassakin alueessa. Valitsemme muuttujan ja jakopisteen saadaksemme parhaan sovituksen. Sitten toinen tai molemmat näistä alueista jaetaan kahteen alueeseen ja tätä prosessia jatketaan, kunnes se jonkun säännön mukaan lopetetaan. Esimerkiksi: Jaetaan alue ensimmäisen kerran kohdasta $X_1 = t_1$. Tämän jälkeen alue $X_1 \leq t_1$ jaetaan kohdasta $X_2 = t_2$, ja alue $X_1 > t_1$ jaetaan kohdasta $X_1 = t_3$. Lopulta alue $X_1 > t_3$ jaetaan kohdasta $X_2 = t_4$. Tämän seurauksena saadaan jako viiteen alueeseen $R_1, R_2, \ldots, R_5$. Vastaava regressiomalli ennustaa vastetta Y vakiolla C_m alueessa R_m, siis

$$\hat{f}(X) = \sum_{m=1}^{5} C_m \mathbb{1}_{\{(X_1, X_2) \in R_m\}}, \tag{2}$$

missä

$$\mathbb{1}_{\{X \in B\}} = \begin{cases} 1, \text{ jos } X \in B \\ 0, \text{ muulloin} \end{cases} \tag{3}$$

on indikaattorifunktio. Malli voidaan esittää binäärisenä puuna.

2.1 Regressiopuut

Siirrymme nyt käsittelemään kysymystä: Kuinka kasvattaa regressiopuu? Oletamme, että data koostuu muuttujista joita on p kpl ja vasteesta, joita on N kpl: Siis (x_i, y_i), $i = 1, \ldots, N$, missä $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$. Algoritmin täytyy päättää automaattisesti muuttujat joiden suhteen jakaa ja jakopisteet, ja myös puun topologia (muoto). Oletamme ensin, että meillä on jako alueisiin $R_1, R_2, \ldots, R_M$. Mallinnamme nyt vastetta vakiolla c_m kussakin alueessa:

$$f(x) = \sum_{m=1}^{M} c_m \mathbb{1}_{(x \in R_m)}. \tag{4}$$

Jos käytämme minimointikriteerinä neliösummaa $\sum (y_i - f(x_i))^2$, niin paras $\hat{c}_m$ on keskimääräinen y_i alueella R_m:

$$\hat{c}_m = \text{ave}(y_i | x_i \in R_m). \tag{5}$$

Perustelut: Määritellään funktio

$$g(c) \coloneqq \sum_{i=1}^{n}(y_i - c)^2 = nc^2 - 2n\bar{y}c + \sum_{i=1}^{n} y_i^2. \tag{6}$$

Nyt $g'' = 2n > 0$, joten funktion g aito globaali minimi löytyy derivaatan nollakohdasta, joka on $\bar{y}$.

Parhaan binäärisen osituksen etsiminen pienimmän neliösumman mielessä on yleisessä tapauksessa laskennallisesti jos ei mahdotonta niin ainakin hyvin epäkäytännöllistä. Etenemme nyt "ahneella"algoritmilla. Tarkastellaan aluksi koko datajoukkoa, muuttujaa j, jonka suhteen ensimmäinen jako tehdään, ja jakopistettä s. Määritellään puolitasot

$$R_1(j, s) = \{X | X_j \leq s\} \quad \text{ja} \quad R_2(j, s) = \{X | X_j > s\}. \tag{7}$$

Etsimme sellaista jakomuuttujaa j ja jakopistettä s jotka ratkaisevat minimointitehtävän

$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right]. \tag{8}$$

Millä tahansa kiinteillä j ja s sulkujen sisällä olevien minimointitehtävien ratkaisut ovat

$$\hat{c}_1 = \text{ave}(y_i | x_i \in R_1(j, s)) \quad \text{ja} \quad \hat{c}_2 = \text{ave}(y_i | x_i \in R_2(j, s)). \tag{9}$$

Kullekin jakomuuttujalle jakopisteen määrääminen voidaan tehdä nopeasti järjestämällä havainnot suuruusjärjestykseen ko. jakomuuttujan suhteen ja käymällä läpi kaikki käytännössä merkitykselliset jakopisteet, joita on yhtä paljon kuin havaintoja. Käymällä tällä tavoin läpi kaikki selittäjämuuttujat parhaan parin (j, s) määrääminen on siis mahdollista. Parhaan jakopisteen löydyttyä ositamme datan kahteen jakopisteen määräämään alueeseen ja toistamme jakoprosessin kullekin tuloksena syntyneelle alueelle ja edelleen uusille alueille jne.

Kuinka isoksi puu pitäisi kasvattaa? Selvästi hyvin suuri puu saattaisi ylisovittua dataan, ja toisaalta liian pieni ei välttämättä tunnista oleellisia rakenteita. Puun koko on mallin kompleksisuutta hallitseva säätöparametri, ja optimaalinen puun koko tulisi valita tilanteen mukaan riippuen datasta. Yksi lähestymistapa olisi jakaa puun oksa vain jos vähennys neliösummassa ylittää tietyn ennalta määrätyn kynnysarvon. Tämä on kuitenkin liian lyhytnäköinen strategia, sillä näennäisesti hyödytön jako saattaa johtaa alemmissa

kerroksissa hyödyllisiin jakoihin.

Suosittu strategia on kasvattaa iso puu T_0 lopettaen jakoprosessi vasta kun jokin minimimäärä solmukohtia (kohtia, joista haarautuu kaksi "oksaa") on saavutettu. Tämän jälkeen isoa puuta karsitaan *kompleksisuushinnan* perusteella. Kuvailemme seuraavaksi karsimisprosessin.

Määrittelemme alipuun $T \subset T_0$ olevan mikä tahansa puu joka voidaan saada karsimalla puuta T_0, siis taittamalla kokoon takaisin (vastakkainen operaatio jakamiselle) jokin määrä sisäisiä solmuja (ei alimpia eli "lehtisolmuja"). Käytämme lehtisolmujen indeksinä kirjainta m, siis (lehti)solmu m vastaa aluetta R_m. Olkoon $|T|$ lehtisolmujen lukumäärä puussa T, ja olkoon N_m lehtisolmussa m olevien havaintojen lukumäärä. Määrittelemme kompleksisuushintakriteerin kaavalla

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|, \tag{10}$$

missä

$$\begin{aligned} \hat{c}_m &= \frac{1}{N_m} \sum_{x_i \in R_m} y_i \\ Q_m(T) &= \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2. \end{aligned} \tag{11}$$

Ajatuksena on löytää, kullekin α, alipuu $T_\alpha \subseteq T_0$ joka minimoi kriteerin $C_\alpha(T)$. Säätöparametri $\alpha \geq 0$ hallitsee kompromissia puun koon ja sen dataan sovittuvuuden välillä. Isot α:n arvot johtavat pienempiin puihin T_α ja päinvastoin. Kuten notaatiokin antaa ymmärtää, valinnalla $\alpha = 0$ ratkaisuna saadaan koko puu T_0. Käsittelemme sopivan α:n valintaa alempana.

Jokaiselle α voidaan näyttää [11], että on olemassa yksikäsitteinen pienin alipuu T_α, joka minimoi kriteerin $C_\alpha(T)$. Löytääksemme tämän puun T_α käytämme *heikoimman lenkin karsintaa*: romahdutamme vaiheittaisesti aina sen (sisäisen) solmun, jonka romahduttaminen tuottaa pienimmän lisäyksen summassa $\sum_m N_m Q_m(T)$, ja jatkamme kunnes olemme päässet takaisin yhden solmun puuhun (juureen). Tämä prosessi antaa äärellisen jonon alipuita ja on mahdollista osoittaa, että tämä jono sisältää puun T_α. Parametri α estimoidaan viisi- tai kymmenkertaisella ristiinvalidoinnilla: Valitsemme estimaatiksi sellaisen $\hat{\alpha}$ joka minimoi ristiinvalidoidun neliösumman. Lopullinen puu on $T_{\hat{\alpha}}$.

2.2 Luokittelupuut

Jos tavoitteena on luokittelu johonkin luokista $1, \ldots, K$, ainoat tarvittavat muutokset puualgoritmissa koskevat jako- ja karsimiskriteerejä. Regressiossa käytimme neliöllisen virheen mittaa (11) karsimisvaiheessa, mutta tämä ei sovi luokittelutilanteessa. Solmussa m, joka edustaa aluetta R_m, ja jossa on N_m havaintoa, olkoon

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} \mathbb{1}_{(y_i = k)} \tag{12}$$

suhteellinen luokkaan k kuuluvien havaintojen lukumäärä solmussa m. Luokittelemme solmun m havainnot luokkaan $k(m) = \arg\max_k \hat{p}_{mk}$, siis suurimpaan luokkaan solmussa m. Solmun epäpuhtauden mittoja $Q_m(T)$ ovat mm.

- Luokitteluvirhe: $\frac{1}{N_m} \sum_{i \in R_m} \mathbb{1}_{(y_i \neq k(m))} = 1 - \hat{p}_{mk(m)}$.
- Gini-indeksi: $\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$.
- Devianssi tai *risti-entropia*: $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$.

Kahden luokan tapauksessa, jos p on suhteellinen osuus toisessa luokassa, nämä kolme mittaa ovat $1 - \max(p, 1-p), 2p(1-p)$ ja $-p \log p - (1-p) \log(1-p)$. Kaikki nämä käyttäytyvät likimäärin samantapaisesti p:n funktiona, mutta ristientropia ja Gini-indeksi ovat differentioituvia ja näin ollen paremmin soveltuvia numeeriseen optimointiin.

Lisäksi ristientropia ja Gini-indeksi ovat herkempiä solmutodennäköisyyksien muutoksille, kuin luokitteluvirhe. Esimerkiksi kahden luokan ongelman tapauksessa, jossa kussakin luokassa on 400 havaintoa (merk. (400, 400)), olettakaamme että yksi jako luo solmut (300, 100) ja (100, 300), ja toinen solmut (200, 400), (200, 0). Molemmissa tapauksissa luokitteluvirhe, eli väärin luokiteltujen havaintojen suhteellinen osuus on 0.25, mutta toinen jako tuottaa puhtaan solmun ja on näin ollen todennäköisesti sen tähden parempi. Sekä Gini-indeksi, että ristientropia ovat matalampia jälkimmäisessä tapauksessa. Tästä syystä puun kasvatuksessa on syytä käyttää joko Gini-indeksiä tai ristientropiaa. Hintakompleksisuuden mukaan karsimisessa voidaan käyttää mitä tahansa näistä kolmesta mitasta, mutta tyypillisesti käytetään luokitteluvirhettä.

Gini-indeksillä on kaksi mielenkiintoista tulkintaa. Sen sijaan, että luokiteltaisiin havainnot enemmistöluokkaan solmun kohdalla, voisimme luokitella

ne luokkaan k todennäköisyydellä $\hat{p}_{mk}$. Nyt tämän säännön opetusvirhe solmussa on $\sum_{k \neq k'} \hat{p}_{mk}\hat{p}_{mk'}$, joka on sama kuin Gini-indeksi. Vastaavasti, jos koodaamme jokaisen luokkaan k kuuluvan havainnon luvuksi 1, ja luokkaan k kuulumattomat havainnot luvuksi 0, solmun varianssi tämän 0-1 vasteen yli on $\hat{p}_{mk}(1-\hat{p}_{mk})$. Summaamalla varianssit kaikkien luokkien yli ($k = 1, \ldots, K$) saadaan Gini-indeksi.

3 Luokittelu- ja regressiopuumallien vahvistaminen

3.1 AdaBoost

Vahvistaminen on voimakkaimpia viimeisen kahdenkymmenen vuoden aikana esitettyjä tilastolliseen oppimiseen liittyviä ajatuksia. Se kehitettiin alun perin luokittelua varten, mutta se on joissain tapauksissa erittäinen hyödyllinen myös regressiomalleja rakennettaessa. Vahvistamisen perusajatus on yhdistää "heikkoja" luokittelijoita vahvan luokittelijoiden "komitean" luomiseksi.

Aloitamme kuvailemalla yhden tunnetuimmista vahvistusalgoritmeista nimeltään AdaBoost. Tarkastellaan binääristä luokittelua, jossa vastemuuttujan mahdolliset luokat ovat -1 ja 1. Annetulla selittäjämuuttujien satunnaisvektorilla X luokittelija $G(X)$ tuottaa ennusteen -1 tai 1. Opetusaineiston suhteellinen virhe on

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{(y_i \neq G(x_i))}, \tag{13}$$

ja suhteellisen virheen odotusarvo on $\text{E}_{XY}\mathbb{1}_{(Y \neq G(X))}$.

Heikko luokittelija on sellainen jonka suhteellinen virhe hieman parempi kuin mitä saataisiin valitsemalla luokka heittämällä lanttia. Vahvistamisen tarkoitus on soveltaa toistuvasti heikkoa luokittelualgoritmia joka askeleella muuttuvaan havaintoaineistoon, ja tällä tavoin tuottaa jono heikkoja luokittelijoita $G_m(x), m = 1, 2, \ldots, M$. Tämän jälkeen näiden heikkojen luokittelijoiden ennusteet yhdistetään painotetusti lopullisen mallin tuottamaksi ennusteeksi:

$$G(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right). \tag{14}$$

Vahvistusalgoritmi määrää painokertoimet $\alpha_1, \alpha_2, \ldots, \alpha_M$, ja sitä kautta

kunkin vastaavan heikon luokittelijan $G_m(x)$ panoksen koko mallissa. Nämä antavat enemmän painoarvoa tarkemmille luokittelijoille.

Joka askeleella algoritmissa painotetaan havaintoja $(x_i, y_i), i = 1, 2, \ldots, N$ painoilla $w_1, w_2, \ldots, w_N$. Alussa painoiksi asetataan $w_i = 1/N, i = 1, 2, \ldots, N$, joten ensimmäisellä askeleella opetetaan luokittelija tavalliseen tapaan alkuperäisellä muokkaamattomalla havaintoaineistolla. Seuraavilla iteraatioilla $m = 2, 3, \ldots, M$ havaintojen painoja muutetaan yksitellen, ja luokittelualgoritmia sovelletaan näihin painotettuihin havaintoihin. Askeleella m ne havainnot, jotka luokittelija $G_{m-1}(x)$ luokitteli väärin saavat aiempaa suuremmat painokertoimet, kun taas oikein painotettujen havaintojen painoja pienennetään. Tällä tavoin vaikeasti luokiteltavat havainnot saavat sitä suuremman painon mitä kauemmin algoritmia ajetaan.

Algoritmi 1 AdaBoost.M1.

1: Alusta havaintojen painot $w_i = 1/N$, $i = 1, \ldots, N$.
2: Jokaiselle m arvoilla $1, \ldots, M$:
   (a) Sovita luokittelija $G_m(x)$ opetusdataan käyttäen painoja w_i.
   (b) Laske
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i \mathbb{1}_{(y_i \neq G_m(x_i))}}{\sum_{i=1}^{N} w_i}.$$
   (c) Laske $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Aseta $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot \mathbb{1}_{(y_i \neq G_m(x_i))}]$, $i = 1, 2, \ldots, N$.
3: Tuloksena saadaan luokittelija $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Algoritmi 1 kuvailee AdaBoost.M1-algoritmin yksityiskohtaisesti. Kullakin askeleella luokittelija $G_m(x)$ opetetaan painotetulla havaintoaineistolla rivillä 2(a). Painotettu suhteellinen virhe lasketaan rivillä 2(b). Rivillä 2(c) lasketaan luokittelijan $G_m(x)$ panoksen lopullisessa mallissa $G(x)$ määräävä painokerroin α_m. Kutakin havaintoa vastaavat painot seuraavaa iteraatiota varten päivitetään rivillä 2(d). Luokittelijan $G_m(x)$ väärin luokittelemien havaintojen painot skaalataan kertoimella $\exp(\alpha_m)$, kasvattaen tällä tavoin niiden suhteellista merkitystä jonossa seuraavan luokittelijan $G_{m+1}(x)$ opettamisessa.

Huomautus 3.1. Heikolle luokittelijalle asetettu vaatimus siitä, että suhteellinen virhe on suurempi kuin $\frac{1}{2}$ johtaa siihen että α_m on positiivinen, mikä johtaa siihen, että $\exp(\alpha_m) > 1$.

Huomautus 3.2. Yleisessä tapauksessa, jossa mahdolliset luokat ovat $\{1, \ldots, J\}$,

havainnolle x lasketaan joka luokalle j äänestystulos

$$s(j,x) = \sum_{m=1}^{M} \alpha_m \mathbb{1}_{(x \in G_m^{-1}(\{j\}))}, \tag{15}$$

ja x luokitellaan suurimman äänestystuloksen saavaan luokkaan

$$\arg\max_j s(j,x). \tag{16}$$

3.2 Vahvistaminen sovittaa additiivisen mallin

Vahvistaminen on tapa sovittaa additiivinen kehitelmä joukkoon yksinkertaisia kantafunktioita. Tässä kantafunktiota ovat luokittelijat $G_m(x) \in \{-1, 1\}$. Yleisemmin kantafunktiokehitelmät ovat muotoa

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m), \tag{17}$$

missä β_m, $m = 1, 2, \ldots, M$ ovat ekspansion kertoimet, ja $b(x;\gamma) \in \mathbb{R}$ ovat tavallisesti yksinkertaisia vektorimuuttujan x ja parametrijoukon γ karakterisoimia funktioita.
Tyypillisesti nämä mallit sovitetaan minimoimalla virhefunktio keskiarvoistettuna opetusdatan yli, kuten neliövirhe- tai uskottavuuspohjainen virhefunktio

$$\min_{\{\beta_m, \gamma_m\}_{m=1}^{M}} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m b(x_i; \gamma_m)\right). \tag{18}$$

Monille virhefunktioille $L(y, f(x))$ ja/tai kantafunkioille $b(x;\gamma)$ tämä vaatii laskennallisesti intensiivisiä numeerisia optimointimenetelmiä. Kuitenkin yksinkertainen vaihtoehto voidaan usein löytää silloin kun on mahdollista ratkaista nopeasti yhden kantafunktion sovittamisen määräämä aliongelma.

3.3 Eteenpäin vaiheittainen additiivinen mallinnus

Eteenpäin vaiheittainen mallinnus approksimoi minimointiongelman (18) ratkaisua lisäämällä vaiheittain ekspansioon uusia kantafunktioita säätämättä jo malliin lisättyjen funktioiden parametreja ja kertoimia. Tämä ajatus on hahmoteltu Algoritmissa 2. Jokaisella iteraatiolla m ratkaistaan optimaalinen kantafunktio $b(x;\gamma_m)$ ja sitä vastaava kerroin β_m, ja termi $\beta_m b(x;\gamma_m)$ lisätään senhetkiseen ekspansioon $f_{m-1}(x)$. Tämä tuottaa päivitetyn ekspansion $f_m(x)$ ja tällä tavoin prosessia jatketaan. Neliöllinen virhe

$$L(y, f(x)) = (y - f(x))^2 \tag{19}$$

on

$$\begin{aligned} L(y_i, f_{m-1}(x_i) + \beta b(x_i;\gamma)) &= (y_i - f_{m-1}(x_i) - \beta b(x_i;\gamma))^2 \\ &= (r_{im} - \beta b(x_i;\gamma))^2, \end{aligned} \tag{20}$$

missä $r_{im} = y_i - f_{m-1}(x_i)$ on mallin $m-1$ virhe eli residuaali havainnon i kohdalla. Siis neliöllisen virheen tapauksessa askeleella m kehitelmään lisätään sellainen termi $\beta_m b(x;\gamma_m)$ joka parhaiten sovittuu mallin $m-1$ residuaaleihin. Tämä idea on pohjana pns-regression vahvistamisessa. Kuitenkin, kuten tulemme myöhemmin näyttämään, neliöllisen virheen minimointi ei yleisesti ole hyvä tapa tehdä luokittelua; siksi meidän on tarkasteltava erilaisia virhekriteerejä.

Algoritmi 2 Eteenpäin vaiheittainen additiivinen mallinnus

1: Alusta $f_0(x) = 0$.
2: Jokaiselle muuttujan m arvolle 1:stä M:ään:

(a) Laske

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i;\gamma)).$$

(b) Aseta $f_m(x) = f_{m-1}(x) + \beta_m b(x;\gamma_m)$.

3.4 Eksponentiaalinen virhe ja AdaBoost

Näytämme nyt, että Algoritmi 1 on ekvivalentti Algoritmin 2 kanssa, jos jälkimmäisessä käytetään virhefunktiota

$$L(y, f(x)) = \exp(-yf(x)). \tag{21}$$

Tämän kriteerin soveltuvuutta käsitellään seuraavassa osiossa.
AdaBoostissa kantafunktiot ovat yksittäiset luokittelijat $G_m(x) \in \{-1, 1\}$. Kun käytetään eksponentiaalista virhefunktiota, joka askeleella on ratkaistava

$$(\beta_m, G_m) = \arg\min_{\beta,G} \sum_{i=1}^{N} \exp[-y_i(f_{m-1}(x_i) + \beta G(x_i))], \tag{22}$$

jonka jälkeen luokittelija G_m kerrottuna kertoimella β_m lisätään senhetkiseen ekspansioon. Tehtävä (22) voidaan esittää muodossa

$$(\beta_m, G_m) = \arg\min_{\beta,G} \sum_{i=1}^{N} w_i^{(m)} \exp(-\beta y_i G(x_i)), \tag{23}$$

missä $w_i^{(m)} = \exp\left(-y_i f_{m-1}(x_i)\right)$. Koska $w_i^{(m)}$ ei riipu kertoimesta β, eikä arvosta $G(x)$, se voidaan ajatella vastaavan havainnon painokertoimena. Tämä painokerroin riippuu arvosta $f_{m-1}(x_i)$, joten yksittäisten painokertoimien arvot muuttuvat joka iteraatiolla.
Tehtävän (23) ratkaisu saavutetaan kaksivaiheisesti. Ensiksi, mielivaltaiselle $\beta > 0$ ratkaisu on

$$G_m = \arg\min_G \sum_{i=1}^{N} w_i^{(m)} \mathbb{1}_{(y_i \neq G(x_i))}. \tag{24}$$

Tämä on luokittelija, joka minimoi painotetun suhteellisen virheen kun ennustetaan luokkaa y. Tämän voi nähdä esittämällä tehtävän (23) minimoitavan osan muodossa

$$e^{-\beta} \cdot \sum_{y_i = G(x_i)} w_i^{(m)} + e^{\beta} \cdot \sum_{y_i \neq G(x_i)} w_i^{(m)},$$

joka voidaan edelleen kirjoittaa muodossa

$$(e^{\beta} - e^{-\beta}) \cdot \sum_{i=1}^{N} w_i^{(m)} \mathbb{1}_{(y_i \neq G(x_i))} + e^{-\beta} \cdot \sum_{i=1}^{N} w_i^{(m)}. \tag{25}$$

Sijoittamalla (24) tehtävään (23) (joka nyt siis minimoidaan enää vain vakion β suhteen) saadaan ratkaisuksi

$$\beta_m = \frac{1}{2} \log \frac{1 - \mathrm{err}_m}{\mathrm{err}_m}, \tag{26}$$

missä err_m on minimoitu painotettu suhteellinen virhe

$$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i^{(m)} \mathbb{1}_{(y_i \neq G(x_i))}}{\sum_{i=1}^{N} w_i^{(m)}}. \tag{27}$$

Tämän jälkeen approksimaatio päivitetään

$$f_m(x) = f_{m-1}(x) + \beta_m G_m(x),$$

minkä johdosta seuraavan iteraation painot ovat

$$w_i^{(m+1)} = w_i^{(m)} \cdot e^{-\beta_m y_i G_m(x_i)}. \tag{28}$$

Koska $-y_i G_m(x_i) = 2 \cdot \mathbb{1}_{(y_i \neq G_m(x_i))} - 1$, painokertoimista (28) tulee

$$w_i^{(m+1)} = w_i^{(m)} \cdot e^{\alpha_m \mathbb{1}_{(y_i \neq G_m(x_i))}} \cdot e^{-\beta_m}, \tag{29}$$

missä $\alpha_m = 2\beta_m$ on Algoritmin 1 kohdassa 2:(c) oleva suure. Tekijä $e^{-\beta_m}$ kaavassa (29) kertoo kaikki painot samalla arvolla, joten sillä ei ole vaikutusta. Näin ollen (29) on ekvivalentti Algoritmin 1 kohdan 2:(d) kanssa. AdaBoost-algoritmin kohdan 2:(a) voi nähdä minimointiongelman (24) ratkaisumetodina. Täten voimme sanoa että Algoritmi 1 minimoi eksponentiaalisen virhekriteerin (21) eteenpäin vaiheittaisella additiivisella mallinnuksella.

3.5 Eksponentiaalisen virheen ominaisuuksista

Pääasiallinen etu eksponentiaalisen virheen käytössä additiivisessa mallinnuksessa on laskennallinen; se johtaa yksinkertaiseen modulaariseen uudelleenpainotusalgoritmiin, nimittäin AdaBoost-algoritmiin. On kuitenkin tarpeen tarkastella sen tilastollisia ominaisuuksia. Mitä ja kuinka hyvin se estimoi? Ensimmäiseen kysymykseen vastataan etsimällä sen populaatiominimoija. Voidaan osoittaa [6], että

$$f^*(x) = \arg\min_{f(x)} E_{Y|x}(e^{-Yf(x)}) = \frac{1}{2}\log\frac{\Pr(Y=1|x)}{\Pr(Y=-1|x)}, \tag{30}$$

tai yhtäpitävästi

$$\Pr(Y=1|x) = \frac{1}{1+e^{-2f^*(x)}} \tag{31}$$

Siis AdaBoostin tuottama additiivinen ekspansio estimoi puolet tapahtumaan $\{Y=1\}$ liittyvän vedonlyöntikertoimen logaritmista.
Toinen saman populaatiominimoijan virhekriteeri on negatiivinen binomiaalinen log-uskottavuus tai *devianssi*. Tässä tulkinnassa f on logit-muunnos. Olkoon

$$p(x) = \Pr(Y=1|x) = \frac{e^{f(x)}}{e^{-f(x)}+e^{f(x)}} = \frac{1}{1+e^{-2f(x)}}. \tag{32}$$

Määritellään lisäksi $Y' = (Y+1)/2 \in \{0,1\}$. Tällöin binominen log-uskottavuusvirhefunktio on

$$l(Y,p(x)) = Y'\log p(x) + (1-Y')\log(1-p(x)), \tag{33}$$

tai yhtäpitävästi devianssi on

$$-2l(Y,f(x)) = 2\log\left(1+e^{-2Yf(x)}\right). \tag{34}$$

Koska log-uskottavuuden populaatiomaksimoija on todellisissa todennäköisyyksissä $p(x) = \Pr(Y=1|x)$, nähdään kohdasta (32), että odotusarvojen $E_{Y|x}[-l(Y,f(x))]$ ja $E_{Y|x}[e^{-Yf(x)}]$ populaatiominimoijat ovat samat. Siis, käyttämällä kumpaa tahansa kriteeriä päädytään samaan ratkaisuun populaatiotasolla. Huomautettakoon tässä, että $e^{-Yf(x)}$ ei ole todellinen log-uskottavuus, koska se ei ole minkään satunnaismuuttujan $Y \in \{-1,1\}$ pistetodennäköisyysfunktion logaritmi.

3.6 Virhefunktiot ja robustisuus

Vaikka sekä eksponentiaalinen, että binominen devianssi tuottavat saman ratkaisun sovellettuna populaatioyhteisjakaumaan, näin ei tapahdu äärellisten havaintoaineistojen kanssa. Molemmat kriteerit ovat väheneviä marginaalin $yf(x)$ funktioita. Luokittelussa (vasteena $-1/1$) marginaalissa on samanlainen rooli kuin residuaaleilla $y - f(x)$ regressiossa. Luokittelusääntö $G(x) = \text{sign}[f(x)]$ johtaa siihen, että havainnot joiden marginaali on positiivinen ($y_i f(x_i) > 0$) luokitellaan oikein, ja havainnot joiden marginaali on negatiivinen ($y_i f(x_i) < 0$) luokitellaan väärin. Päätöspinta on joukko $\{x \,|\, f(x) = 0\}$. Luokittelun tavoitteena on tuottaa positiivisia marginaaleja niin usein kuin mahdollista. Minkä tahansa luokittelussa käytettävän virhekriteerin tulisi sakottaa negatiivisia marginaaleja raskaammin kuin positiivisia sillä positiivisen marginaalin havainnot on jo luokiteltu oikein.
Missä tahansa vaiheessa opetusprosessia eksponentiaalinen virhekriteeri antaa muita paljon enemmän painoa havainnoille, joilla on itseisarvoltaa suuri negatiivinen marginaali. Binomiaalinen devianssi antaa suhteessa vähemmän painoa tällaisille havainnoille jakaen painotukset tasaisemmin koko datan suhteen. Näin ollen se on paljon robustimpi kohinaisissa asetelmissa, missä Bayesin suhteellinen virhe ei ole lähellä nollaa, ja erityisesti tilanteissa, joissa osa opetusdatan luokkainformaatiosta on väärää. AdaBoostin suorituskyvyn on empiirisesti huomattu huononevan tällaisissa tilanteissa dramaattisesti.
Neliöllisen virheen populaatiominimoija on

$$f^*(x) = \arg\min_{f(x)} E_{Y|x}(Y - f(x))^2 = E(Y|x) = 2 \cdot \Pr(Y = 1|x) - 1. \quad (35)$$

Luokittelusääntönä on edelleen $G(x) = \text{sign}[f(x)]$. Neliöllinen virhe ei ole hyvä väärinluokitteluvirheen sijainen. Se ei ole vähenevä marginaalin $yf(x)$ funktio. Marginaalin arvoilla $y_i f(x_i) > 1$ se kasvaa neliöllisesti antaen kasvavan painon (virheen) havainnoille jotka luokitellaan oikein kasvavalla varmuudella, ja tällä tavoin vähentäen väärin luokiteltujen havaintojen (joille $y_i f(x_i) < 0$) suhteellista painoa. Siis, jos tavoitteena on luokittelu, vähenevä kriteeri täyttää paremmin virhefunktion paikan.

Tarkastellaan nyt tilanteita, joissa luokkia on $K > 2$ kpl, ja vaste Y saa siis arvoja järjestämättömässä joukossa kategorioita $\mathcal{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_K\}$. Etsimme nyt luokittelijaa $G(x)$ joka saa arvoja joukossa $\mathcal{G}$. Tässä tilanteessa riittää ehdollisten luokkatodennäköisyyksien $p_k(x) = \Pr(Y = \mathcal{G}_k|x)$, $k = 1, \ldots, K$, tunteminen, sillä tällöin Bayesin luokittelija on

$$G(x) = \mathcal{G}_k, \quad k = \arg\max_{\ell} p_\ell(x). \quad (36)$$

Periaatteessa luokkatodennäköisyyksiä $p_k(x)$ ei tarvitse oppia, vaan ainoastaan tieto siitä, mikä luokkatodennäköisyyksistä on suurin. Kuitenkin datanlouhintasovelluksissa kiinnostus on usein enemmän varsinaisissa luokkatodennäköisyyksissä $p_\ell(x)$, $\ell = 1, \ldots, K$, kuin luokittelussa. *Logistinen* malli yleistyy luonnollisesti K:n luokan tapaukseen:

$$p_k(x) = \frac{e^{f_k(x)}}{\sum_{l=1}^{K} e^{f_l(x)}}, \tag{37}$$

jolloin $0 \leq p_k(x) \leq 1$ ja todennäköisyyksien summa on 1. Funktioissa $f_k(x)$ on päällekkäisyyttä, sillä mielivaltaisen $h(x)$ lisääminen kuhunkin funktioon jättää mallin ennalleen. Perinteisesti yksi niistä asetetaan nollaksi: esimerkiksi $f_K(x) = 0$. Haluamme tässä säilyttää symmetrian, ja asetamme rajoitteen $\sum_{k=1}^{K} f_k(x) = 0$. Binomiaalinen devianssi yleistyy luonnollisella tavalla K:n luokan *multinomi*devianssivirhefunktioksi:

$$\begin{aligned} L(y, p(x)) &= -\sum_{k=1}^{K} \mathbb{1}_{(y=\mathcal{G}_k)} \log p_k(x) \\ &= -\sum_{k=1}^{K} \mathbb{1}_{(y=\mathcal{G}_k)} f_k(x) + \log \left(\sum_{\ell=1}^{K} e^{f_\ell(x)} \right). \end{aligned} \tag{38}$$

Kuten kahden luokan tapauksessa, kriteeri (38) sakottaa virheellisiä ennusteita vain lineaarisesti niiden virheellisyyden asteen mukaan. Ei tunneta luonnollista tapaa yleistää eksponentiaalinen virhekriteeri K:n luokan tapaukseen.

3.7 Robustit virhefunktiot regressiossa

Regressioasetelmassa neliöllisen virheen $L(y, f(x)) = (y - f(x))^2$ ja absoluuttisen virheen $L(y, f(x)) = |y - f(x)|$ suhde on analoginen eksponentiaalisen virheen ja binomiaalisen log-uskottavuuden suhteeseen. Populaatioratkaisut ovat nyt $f(x) = E(Y|x)$ neliölliselle virheelle, ja $f(x) = \text{median}(Y|x)$ absoluuttiselle virheelle; symmetristen virhejakaumien tapauksessa nämä ovat samat. Kuitenkin äärellisillä otoksilla neliöllinen virhe antaa sovitusvaiheessa paljon enemmän painoa havainnoille, joilla on suuret residuaalien itseisarvot $|y_i - f(x_i)|$. Se on näin ollen huomattavasti vähemmän robusti, ja sen suorituskyky heikkenee huomattavasti pitkähäntäisten virhejakaumien tapauksessa ja erityisesti (merkittävän) virheellisesti kirjattujen y-arvojen tapauksessa. Muita robustimpeja kriteerejä, kuten absoluuttinen virhe suoriutuvat paljon paremmin näissä tilanteissa. Tilastollisessa robustisuuskirjallisuudessa on ehdotettu monenlaisia regressiovirhekriteerejä oudokeilta ja vääriltä kirjauksilta suojautumisen varalle, jotka samaan aikaan ovat lähes yhtä tehokkaita

kuin PNS normaalijakautuneille virheille. Ne ovat usein sekä neliöllistä - että absoluuttista virhettä parempia suhteellisen paksuhäntäisten virhejakaumien tapauksessa. Yksi tällainen kriteeri on *Huberin kriteeri M-regressiossa*

$$L(y, f(x)) = \begin{cases} [y - f(x)]^2, & \text{kun } |y - f(x)| \leq \delta \\ \delta(|y - f(x)| - \delta/2), & \text{muulloin.} \end{cases} \tag{39}$$

Absoluuttinen virhe regressiossa on analoginen binomiaalisen devianssin kanssa luokittelussa: Se kasvaa lineaarisesti äärimmäisten marginaalien kasvaessa (äärimmäisten marginaalien funktiona). Eksponentiaalinen virhe on neliöllistäkin virhettä rajumpi: Se sakottaa eksponentiaalisesti, ei neliöllisesti.

Edellä esitetyn perusteella neliöllinen virhe regressiossa ja eksponentiaalinen virhe luokittelussa eivät ole tilastollisesta perspektiivistä parhaita kriteerejä silloin, kun mallilta vaaditaan robustisuutta, kuten erityisesti datanlouhintasovelluksissa asianlaita on. Ne molemmat voivat kuitenkin johtaa elegantteihin modulaarisiin vahvistusalgoritmeihin eteenpäin vaiheittaisen additiivisen mallinnuksen kontekstissa. Neliöllisen virheen tapauksessa yksinkertaisesti sovitetaan kantaoppija senhetkisen mallin residuaaleihin $y_i - f_{m-1}(x_i)$ joka askeleella. Eksponentiaalisen virheen tapauksessa suoritetaan kantaoppijan painotettu sovitus mitattuihin vasteisiin y_i, painoina $w_i = \exp(-y_i f_{m-1}(x_i))$. Käyttämällä suoraan näiden paikalla muita, robustimpia kriteerejä tuloksena ei saada näin yksinkertaisia vahvistusalgoritmeja. Kuitenkin minkä tahansa differentioituvan virhekriteerin perusteella on mahdollista johtaa yksinkertaisia elegantteja vahvistusalgoritmeja. Tällä tavoin voidaan tuottaa datanlouhintasovelluksia varten hyvin robusteja vahvistusalgoritmeja.

3.8 Marginaali ja reuna

Alikappaleissa 3.1 - 3.5 AdaBoost esitetään vaiheittaisen additiivisen mallinnuksen kontekstissa, tarkemmin sanottuna sellaisena erikoistapauksena vaiheittaisesta additiivisesta mallinnuksesta, missä tappiofunktioksi on valittu eksponentiaalinen virhe.

AdaBoostille ja yleisemmin vahvistamiselle on tarjottu muunkinlaisia tulkintoja. Schapire, Freund, Bartlett ja Lee esittävät artikkelissaan [12] että vahvistamismenetelmät kasvattavat *marginaalia*, minkä takia yleistysvirhe voi laskea vielä senkin jälkeen, kun opetusvirhe on painunut nollaan. Huomautettakoon tässä että kappaleessa 3.8 määrittelemme marginaalin eri tavalla kuin kappaleessa 3.6. Breiman kehittelee osittain tämän pohjalta omaa selitystään AdaBoostin toimivuudelle [2]. Hänen tulkintansa mukaan Ada-

Boost pienentää testivirhettä pienentämällä niin sanottua *reunaa*, missä reuna tarkoittaa kaikkien virheellisen luokituksen antaneiden heikkojen luokittelijoiden "äänien" summaa. Esittelemme seuraavaksi Breimanin kehittelemää teoriaa AdaBoost-algoritmista ja vahvistamisesta.

Olkoon $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ N:n vektorin avaruus, missä vektorit $\mathbf{x}_i,\ i = 1, \ldots, N$ ovat selittäjämuuttujia joita vastaavat luokat ovat $y_1, \ldots, y_N$. Olkoon H sellainen luokittelijoiden $\{h_1(\mathbf{x}), \ldots, h_M(\mathbf{x})\}$ komitea, että jokainen funktioista $h_m(\mathbf{x}),\ m = 1, \ldots, M$ saa arvoja joukossa $\{1, \ldots, J\}$. Oletetaan lisäksi, että havainnot $\mathbf{x}_i,\ i = 1, \ldots, N$ noudattavat sellaista todennäköisyysjakaumaa P, että $P(\mathbf{x}_n) > 0$ kaikilla n. Jos muuttujan $\mathbf{x}_n$ todellinen luokka on y_n, kutsumme joukkoa $e_m = \{\mathbf{x}_n;\ h_m(\mathbf{x}_n) \neq y_n\}$ luokittelijan h_m virhejoukoksi. Lopullinen luokittelija on

$$\begin{aligned} G(x) &= \arg\max_j s(j, \mathbf{x}) = \arg\max_j \sum_{m=1}^{M} \alpha_m \mathbb{1}_{(\mathbf{x} \in h_m^{-1}(\{j\}))} \\ &= \arg\max_j \sum_{m:h_m(\mathbf{x})=j} \alpha_m, \end{aligned} \tag{40}$$

missä α_m on luokittelijan h_m epänegatiivinen äänen suuruus. Äänille pätee $\sum_{m=1}^{M} \alpha_m = 1$, ja $s(j, \mathbf{x})$ on havainnolle $\mathbf{x}$ luokan j saaman äänimäärän suuruus.

Määritelmä 3.3. Jos havainnon $\mathbf{x}$ todellinen luokka on j, niin määrittelemme havaintoon $\mathbf{x}$ ja luokkaan j liittyvän *marginaalin*

$$\text{margin}(\boldsymbol{\alpha}, \mathbf{x}) = s(j, \mathbf{x}) - \max(s(i, \mathbf{x});\ i \neq j), \tag{41}$$

missä $s(j, \mathbf{x})$ tulee kaavasta (15).

Marginaali on siis oikean luokan äänien ja eniten ääniä saaneen virheellisen luokan äänien erotus. Näin ollen virheellisen luokittelun todennäköisyys, kun jätämme tasatilanteet huomiotta, on $P(\text{margin}(\boldsymbol{\alpha}, \mathbf{x}) < 0)$.

Määritelmä 3.4. Olkoon $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_M)$ vektori epänegatiivisia numeroita (ääniä), joille pätee $\sum_{m=1}^{M} \alpha_m = 1$. Määritellään muuttujan $\mathbf{x}$ funktio *reuna* konveksina kombinaationa:

$$\text{edge}(\boldsymbol{\alpha}, \mathbf{X}) = \sum_{m=1}^{M} \alpha_m \mathbb{1}_{\{\mathbf{X} \in e_m\}}, \tag{42}$$

kun $\mathbf{X}$ on satunnainen ja

$$\text{edge}(\boldsymbol{\alpha}, \mathbf{x}) = \sum_{m=1}^{M} \alpha_m \mathbb{1}_{(\mathbf{x} \in e_m)} \tag{43}$$

jollekin tietylle $\mathbf{x}$. Jatkossa muuttuja $\mathbf{x}$ tai $\mathbf{X}$ jätetään usein merkitsemättä. Se, onko kyseessä satunnaissuure, selviää asiayhteydestä.

Huomautus 3.5. Reunan ja marginaalin välillä seuraava yhteys:

$$\begin{aligned} \text{margin}(\boldsymbol{\alpha}, \mathbf{x}) &= \sum_{m:h_m(\mathbf{x})=y} \alpha_m - \max_{y' \neq y} \sum_{m:h_m(\mathbf{x})=y'} \alpha_m \\ &= 1 - \max_{y' \neq y} \sum_{m:h_m(\mathbf{x})=y'} \alpha_m - \sum_{m:h_m(\mathbf{x}) \neq y} \alpha_m \\ &\geq 1 - 2 \sum_{m:h_m(\mathbf{x}) \neq y} \alpha_m = 1 - 2 \cdot \text{edge}(\boldsymbol{\alpha}, \mathbf{x}). \end{aligned} \tag{44}$$

Kahden luokan tapauksessa

$$\begin{aligned} &\max_{y' \neq y} \sum_{m:h_m(\mathbf{x})=y'} \alpha_m = \sum_{m:h_m(\mathbf{x}) \neq y} \alpha_m \\ &\implies \text{margin}(\boldsymbol{\alpha}, \mathbf{x}) = 1 - 2 \cdot \text{edge}(\boldsymbol{\alpha}, \mathbf{x}). \end{aligned} \tag{45}$$

Seuraus 3.6. *Kaikilla $\theta \in (-1, 1)$ pätee*

$$P(\text{margin}(\boldsymbol{\alpha}) < \theta) \leq P(\text{edge}(\boldsymbol{\alpha}) > \frac{1-\theta}{2}). \tag{46}$$

Todistus. Kaavan (44) perusteella

$$\begin{aligned} &\{\omega \in \Omega : \text{margin}(\omega) < \theta\} = \{\omega \in \Omega : \frac{1-\theta}{2} < \frac{1 - \text{margin}(\omega)}{2}\} \\ &\subseteq \{\omega \in \Omega : \frac{1-\theta}{2} < \text{edge}(\omega)\}, \end{aligned} \tag{47}$$

joten todennäköisyysmitan ominaisuuksien nojalla

$$P(\{\omega \in \Omega : \text{margin}(\omega) < \theta\}) \leq P\left(\{\omega \in \Omega : \frac{1-\theta}{2} < \text{edge}(\omega)\}\right). \tag{48}$$

□

Tästä Breiman päättelee, että sellaiset kerroinvektorit $\boldsymbol{\alpha}$, joiden reuna on pieni, johtavat pieneen yleistysvirheeseen. Esim. Jos opetusdatassa on harvinaista, että yli $\frac{1}{4}$ luokittelijoiden komitean äänistä ovat virheellisiä, eli että todennäköisyys $P(\text{edge}(\boldsymbol{\alpha}) > \frac{1}{4})$ on pieni, niin todennäköisyys, että kohdepopulaatiossa yli $\frac{1}{2}$ painotetuista äänistä on väärin ($P(\text{margin}(\boldsymbol{\alpha}) < \frac{1}{2})$) on pieni [2].

Tavoitteena on nyt annetulla luokittelijoiden joukolla H, ja luokilla $\{y_1, \ldots, y_N\}$, löytää sellainen kerroinvektori $\boldsymbol{\alpha}$ joka minimoi todennäköisyyden $P(\text{edge}(\boldsymbol{\alpha}) > \phi)$ pienillä ϕ:n arvoilla. Määritelemme *alareunan* ϕ^* pienimmäksi sellaiseksi ϕ, jolle

$$\inf_{\boldsymbol{\alpha}} P(\text{edge}(\boldsymbol{\alpha}) > \phi) = 0. \tag{49}$$

Todennäköisyyden $P(\text{edge}(\boldsymbol{\alpha}) > \phi)$ tai yleisemmin odotusarvon $E[f(\text{edge}(\boldsymbol{\alpha}))]$, missä f on kasvava funktio, minimoimiseksi on olemassa algoritmeja. Esimerkiksi todennäköisyytta $P(\text{edge}(\boldsymbol{\alpha}) > \phi)$ voidaan minimoida lineaarisen optimoinnin menetelmillä. Yleensä algoritmit jotka suppenevat sellaiseen vektoriin $\boldsymbol{\alpha}$, että $P(\text{edge}(\boldsymbol{\alpha}) > \phi) = 0$ käyttävät etukäteen määrättyä arvoa vakiolle ϕ. On olemassa myös *universaaleja* algoritmeja, jotka suppenevat sellaista arvoa $\boldsymbol{\alpha}$ kohti, että $P(\text{edge}(\boldsymbol{\alpha}) > \phi)$ kaikilla $\phi > \phi^*$.

Lause 3.7. *$\phi^* = \max_Q \min_m Q(e_m)$, missä maksimi otetaan kaikkien avaruuden $\mathcal{X}$ todennäköisyyksien Q yli.*

3.8.1 Mukautuvat algoritmit

Funktion $Ef(edge(\boldsymbol{\alpha}))$ minimoimiseksi, ehdoilla $\alpha_m \geq 0$ kaikilla $m = 1, \ldots, M$ ja $\sum_{m=1}^{M} \alpha_m = 1$, voidaan muodostaa monia algoritmeja. Tarkastelemme nyt niin kutsuttuja *mukautuvia uudelleenpainotusalgoritmeja* (adaptive reweighting and combining algorithms), lyh. mukautuvia algoritmeja, jotka käyttävät mukautuvaa uudelleenpainotusta luokittelijan ja sen äänen valinnassa.
Oletamme sellaisen iteratiivisen algoritmin, joka toistaa seuraavat vaiheet joka askeleella:

(i) Algoritmi määrittelee todennäköisyyden $Q_k(\mathbf{x}_n)$ riippuen opetusdatan luokitteluvirheistä ja siihen mennessä valitun $k-1$:n ensimmäisen luokittelijan äänistä.

(ii) k:s valittu luokittelija on h_{m^*}, missä

$$m^* = \arg\min_m Q_k(e_m)$$

(iii) Luokittelijan h_{m^*} ääni α_{m^*} määräytyy sen opetusdatassa tekemistä luokitteluvirheistä ja $k-1$:n ensimmäisen luokittelijan tekemistä virheistä opetusdatassa ja niiden äänistä.

Rajoitus (ii) on kaikista hyödyllisin ja mielenkiintoisin. Se tarkoittaa että annetulla Q valitsemme sellaisen luokittelijan h_m luokasta H, jolla on pienin Q-virhetodennäköisyys. Jos luokka H on ääretön, siitä ei ole tavallisesti käytännössä mahdollista valita pienimmän Q-virheen luokittelijaa. Sitä voidaan kuitenkin approksimoida rajoittumalla johonkin äärelliseen luokittelijoiden luokkaan.

3.8.2 Normalisoimattomat mukautuvat algoritmit

Olkoon $f\colon \mathbb{R} \to \mathbb{R}$ sellainen funktio, että $f(x) \to \infty$ kun $x \to \infty$, ja $f(x) \to 0$ kun $x \to -\infty$. Oletetaan lisäksi että funktiolla f on kaikkialla positiiviset ensimmäisen ja toisen kertaluvun derivaatat. Merkitään epänegatiivisille painoille $\{\mathrm{b}_m\}$ $(\mathbf{b}, \mathbb{1}) = \sum_{m=1}^{M} b_m \mathbb{1}_{\{\mathbf{x} \in e_m\}}$, $(\mathbf{b}, \mathbb{1}(\mathbf{x}_n)) = \sum_{m=1}^{M} b_m \mathbb{1}_{(\mathbf{x}_n \in e_m)}$ ja $|\mathbf{b}| = \sum_{m=1}^{M} b_m$. Olettaen $\phi > \phi^*$, lähdetään minimoimaan funktiota $g(\mathbf{b}) = Ef((\mathbf{b}, \mathbb{1}) - \phi|\mathbf{b}|)$ lähtien arvosta $\mathbf{b} = 0$.

Normalisoimaton algoritmi

Algoritmissa 3 $\mathbf{u}_{m^*}$ tarkoittaa yksikkövektoria $(0, 0, \ldots, 0, 1, 0, 0, \ldots, 0)$, jossa m^*:s alkio on yksi, ja $b_\ell, \ell = 1, \ldots, L$, ovat kerroinvektorin $\mathbf{b}$ nollasta poikkeavat komponentit.

Huomautus 3.8. Algoritmin 3 antamassa luokittelijassa on mukana korkeintaan K kpl luokittelijoita joukosta H, sillä kerroinvektorilla $\mathbf{b}$ on korkeintaan K nollasta poikkeavaa komponenttia.

Huomautus 3.9. Minimointitehtävällä (52) on yksikäsitteinen ratkaisu, sillä

$$\frac{\mathrm{d}^2 g(\mathbf{b} + \Delta \mathbf{u}_{m^*})}{\mathrm{d}\Delta^2} = E\left[(\mathbb{1}_{\{\mathbf{x} \in e_{m^*}\}} - \phi)^2 f''\left(\sum_{m=1}^{M} b_m \mathbb{1}_{\{\mathbf{x} \in e_m\}} - \phi \sum_{m=1}^{M} b_m + \Delta(\mathbb{1}_{\{\mathbf{x} \in e_{m^*}\}} - \phi)\right)\right] > 0. \tag{54}$$

Algoritmi 3 Normalisoimaton mukautuva algoritmi

1: Aseta $\mathbf{b}^{(0)} = 0$.

2: Jokaiselle k arvoilla $1, \ldots, K$:

(a) Aseta

$$Q_k(\mathbf{x}_n) = \frac{f'((\mathbf{b}, \mathbb{1}(\mathbf{x}_n)) - \phi|\mathbf{b}|)}{\sum_{\ell=1}^{N} f'((\mathbf{b}, \mathbb{1}(\mathbf{x}_\ell)) - \phi|\mathbf{b}|)}, \tag{50}$$

$n = 1, \ldots, N$.

(b) ratkaise

$$m^* = \arg\min_m Q_k(e_m) = \arg\min_m \sum_{n=1}^{N} Q_k(\mathbf{x}_n) \mathbb{1}_{(h_m(\mathbf{x}_n) \neq y_n)}, \tag{51}$$

(c) ratkaise

$$\Delta^* = \arg\min_{\Delta > 0} g(\mathbf{b} + \Delta \mathbf{u}_{m^*}), \tag{52}$$

(d) aseta $\mathbf{b}^{(k)} \leftarrow \mathbf{b}^{(k-1)} + \Delta^* \mathbf{u}_{m^*}$.

3: Tuloksena saadaan luokittelija

$$G(\mathbf{x}) = \sum_{m=1}^{M} b_m h_m(\mathbf{x}) = \sum_{\ell=1}^{L} b_\ell h_\ell(\mathbf{x}). \tag{53}$$

Lause 3.10. *Olkoon* $\mathbf{b}^{(k)}$ *normalisoimattoman mukautuvan algoritmin tuottama jono peräkkäisiä arvoja, ja asetetaan* $\mathbf{c}^{(k)} = \frac{\mathbf{b}^{(k)}}{|\mathbf{b}|}$. *Tällöin jos* $\mathbf{c}$ *on jonon* $\mathbf{c}^{(k)}$ *kasautumispiste, niin* $P(\text{edge}(\mathbf{c}) > \phi) = 0$.

Todistus. Väitteen todistamiseksi riittää osoittaa, että $|\mathbf{b}^{(k)}| \to \infty$, sillä kirjoittamalla

$$(\mathbf{b}^{(k)}, \mathbb{1}) - \phi|\mathbf{b}^{(k)}| = |\mathbf{b}^{(k)}|(\text{edge}(\mathbf{c}^{(k)}) - \phi) \tag{55}$$

nähdään, että ellei $P(\text{edge}(\mathbf{c}^{(k)}) > \phi)$ lähesty nollaa, niin $g(\mathbf{b}^{(k)}) \to \infty$. Jos $|\mathbf{b}^{(k)}|$ ei lähesty ääretöntä, sillä on kasautumispiste $\mathbf{b}^*$. Mutta aina, kun $\mathbf{b}^{(k)}$ on kasautumispisteensä $\mathbf{b}^*$ läheisyydessä, $g(\mathbf{b}^{(k)})$ vähenee seuraavalla askeleella vähintään jonkin positiivisen vakion $\delta > 0$ verran. Koska $g(\mathbf{b}^{(k)})$ on vähenevä ja epänegatiivinen, tämä on mahdotonta. □

Huomautus. (i) Tämän perusteella on selvää, että karkeammatkin algoritmit suppenesivat, sillä sitä varten riittää, että tuotetaan sellainen jono $|\mathbf{b}^{(k)}| \to \infty$, että $g(\mathbf{b}^{(k)})$ on rajoitettu. Erityisesti haku suoraa pitkin voidaan välttää.

(ii) Jos oletamme, että $g(0) = 1$, niin askeleella k

$$P(\text{edge}(\mathbf{c}^{(k)}) > \phi) \leq g(\mathbf{b}^{(k)}).$$

Todistus. Oletuksen perusteella

$$g(0) = \sum_{n=1}^{N} Q(\mathbf{x}_n) f(0) = f(0) = 1, \tag{56}$$

joten

$$\begin{aligned} g(\mathbf{b}^{(k)}) &= \sum_{n=1}^{N} Q(\mathbf{x}_n) f\left(\sum_{m=1}^{M} b_m^{(k)} \mathbb{1}_{(\mathbf{x}_n \in e_m)} - \phi \sum_{m=1}^{M} b_m^{(k)}\right) \\ &= \sum_{n=1}^{N} Q(\mathbf{x}_n) \mathbb{1}_{(\sum_{m=1}^{M} \mathbf{c}_m^{(k)} \mathbb{1}_{(\mathbf{x}_n \in e_m)} > \phi)} \underbrace{f\left(\sum_{m=1}^{M} b_m^{(k)} \mathbb{1}_{(\mathbf{x}_n \in e_m)} - \phi \sum_{m=1}^{M} b_m^{(k)}\right)}_{>1} \\ &\quad + \sum_{n=1}^{N} Q(\mathbf{x}_n) \mathbb{1}_{(\sum_{m=1}^{M} \mathbf{c}_m^{(k)} \mathbb{1}_{(\mathbf{x}_n \in e_m)} \leq \phi)} \underbrace{f\left(\sum_{m=1}^{M} b_m^{(k)} \mathbb{1}_{(\mathbf{x}_n \in e_m)} - \phi \sum_{m=1}^{M} b_m^{(k)}\right)}_{>0} \\ &\geq \sum_{n=1}^{N} Q(\mathbf{x}_n) \mathbb{1}_{(\sum_{m=1}^{M} \mathbf{c}_m^{(k)} \mathbb{1}_{(\mathbf{x}_n \in e_m)} > \phi)} = P(\text{edge}(\mathbf{c}^{(k)}) > \phi), \end{aligned} \tag{57}$$

sillä niillä n, joilla $\sum_{m=1}^{M} \mathbf{c}_m^{(k)} \mathbb{1}_{(\mathbf{x}_n \in e_m)} > \phi$, on voimassa

$$f\left(\sum_{m=1}^{M} b_m^{(k)} \mathbb{1}_{(\mathbf{x}_n \in e_m)} - \phi \sum_{m=1}^{M} b_m^{(k)}\right) > 1, \tag{58}$$

koska

$$\sum_{m=1}^{M} b_m^{(k)} \mathbb{1}_{(\mathbf{x}_n \in e_m)} - \phi \sum_{m=1}^{M} b_m^{(k)} > 0, \tag{59}$$

ja koska funktion f ensimmäisen kertaluvun derivaatta on kaikkialla positiivinen. □

(iii) Yksinkertaisin normalisoimattoman algoritmin ehdot täyttävä funktio on e^x. Kutsumme tähän funktioon perustuvia normalisoimattomia algoritmeja *eksponenttifunktioon perustuviksi mukautuviksi algoritmeiksi.*

AdaBoost on eksponenttifunktioon perustuva mukautuva algoritmi, jolle $\phi = \frac{1}{2}$

Kun $f(x) = e^x$, niin

$$f((\mathbf{b}, \mathbf{i}) - \phi|\mathbf{b}|) = e^{-\phi|\mathbf{b}|} \prod_{m=1}^{M} e^{b_m \mathbb{1}_{\{\mathbf{x} \in e_m\}}}. \tag{60}$$

Merkitään $\pi(n) = \prod_{m=1}^{M} e^{b_m \mathbb{1}_{(\mathbf{x}_n \in e_m)}}$, asetetaan $Q(\mathbf{x}_n) = \frac{\pi(n)}{\sum_{\ell=1}^{N} \pi(\ell)}$, ja $m^* = \arg\min_m Q(e_m)$. Asetetaan $\epsilon_m = Q(e_{m^*})$. Haku suoraa pitkin tehdään asettamalla

$$E(\mathbb{1}_{\{\mathbf{x} \in e_m\}} - \phi) f'((\mathbf{b} + \Delta \mathbf{u}_{m^*}, \mathbb{1}) - \phi|\mathbf{b}| - \phi\Delta) = 0. \tag{61}$$

Tämän ratkaisu on $\Delta^* = \log(\frac{\phi}{1-\phi}) + \log(\frac{1-\epsilon_m}{\epsilon_m})$. Jakaumaa Q päivitetään $\pi(n) \to \pi(n) \exp(\Delta^* \mathbb{1}_{(\mathbf{x}_n \in e_{m^*})})$. Kun $\phi = \frac{1}{2}$, tämä on aiemmin kuvailtu AdaBoost-algoritmi.

3.8.3 Normalisoidut mukautuvat algoritmit

Normalisoidut mukautuvat algoritmit käsittelevät suoraan suuretta $\frac{(\mathbf{b}, \mathbb{1})}{|\mathbf{b}|}$, joten kertoimet normalisoidaan summautumaan yhdeksi joka askeleella. Algoritmit minimoivat odotusarvoa $Ef(\frac{(\mathbf{b}, \mathbb{1})}{|\mathbf{b}|})$, missä f' on epänegatiivinen, ja f'' jatkuva ja epänegatiivinen kaikilla $x \in [0, 1]$. Merkitään seuraavassa $\mathbf{c} = \frac{\mathbf{b}}{|\mathbf{b}|}$.

Algoritmi 4 Normalisoitu mukautuva algoritmi

1: Aseta $\mathbf{b}^{(0)} = 0$.

2: Jokaiselle k arvoilla $1, \ldots, K$: Jos $Ef'((\mathbf{c}, \mathbb{1})) = 0$, lopeta. Muussa tapauksessa:

(a) Aseta

$$Q_k(\mathbf{x}_n) = \frac{f'((\mathbf{b}, \mathbb{1}(\mathbf{x}_n))/|\mathbf{b}|)}{\sum_{\ell=1}^{N} f'((\mathbf{b}, \mathbb{1}(\mathbf{x}_\ell))/|\mathbf{b}|)}, \tag{62}$$

$n = 1, \ldots, N$.

(b) Ratkaise

$$m^* = \arg\min_m Q_k(e_m) = \arg\min_m \sum_{n=1}^{N} Q_k(\mathbf{x}_n) \mathbb{1}_{(h_m(\mathbf{x}_n) \neq y_n)}. \tag{63}$$

(c) Jos $Q(e_{m^*}) \geq E_Q(\mathbf{c}, \mathbf{i})$, lopeta. Muussa tapauksessa ratkaise

$$\Delta^* = \arg\min_{\Delta > 0} g(\mathbf{b} + \Delta \mathbf{u}_{m^*}), \tag{64}$$

ja

(d) aseta $\mathbf{b}^{(k)} \leftarrow \mathbf{b}^{(k-1)} + \Delta^* \mathbf{u}_{m^*}$.

3: Tuloksena saadaan luokittelija

$$G(\mathbf{x}) = \sum_{m=1}^{M} b_m h_m(\mathbf{x}) = \sum_{\ell=1}^{L} b_\ell h_\ell(\mathbf{x}). \tag{65}$$

Huomautus. Huomattakoon, että

$$\frac{\partial Ef((\mathbf{b},\mathbb{1})/|\mathbf{b}|)}{\partial b_m} = \frac{1}{|\mathbf{b}|}E(\mathbb{1}_{\{\mathbf{x}\in e_m\}} - (\mathbf{c},\mathbb{1}(\mathbf{x}_n)))f'((\mathbf{b},\mathbb{1})/|\mathbf{b}|). \tag{66}$$

Pienin osittaisderivaatta on kohdassa $m = m^*$.

Lause 3.11. *Olkoon* $\mathbf{c}$ *normalisoidun mukautuvan algoritmin rajapiste. Tällöin* $\mathbf{c}$ *on odotusarvon* $Ef(\text{edge}(\mathbf{c}))$ *globaali minimi.*

Todistus. Olkoon $f'(x) > 0$ kaikilla $x > \phi$, ja $f'(x) = 0$ kaikilla $x \leq \phi$. Jos $\phi < \phi^*$ tai $f'(x) > 0$ kaikilla x, niin ei ole mahdollista että $Ef'((\mathbf{c},\mathbf{i})) = 0$. Oletetaan että algoritmi päättyy kun on otettu äärellinen määrä askelia, koska $Q(e_{m^*}) \geq E_Q(\mathbf{c},\mathbf{i})$. Tällöin

$$E\left(\mathbb{1}_{\{\mathbf{x}\in e_{m^*}\}}f'\left(\frac{(\mathbf{b},\mathbb{1})}{|\mathbf{b}|}\right)\right) \geq \sum_{m=1}^{M} c_m E\left(\mathbb{1}_{\{\mathbf{x}\in e_m\}}f'\left(\frac{(\mathbf{b},\mathbb{1})}{|\mathbf{b}|}\right)\right). \tag{67}$$

Tästä seuraa, että kaikilla m joko $c_m = 0$ tai

$$E\left(\mathbb{1}_{\{\mathbf{x}\in e_{m^*}\}}f'\left(\frac{(\mathbf{b},\mathbb{1})}{|\mathbf{b}|}\right)\right) = E\left(\mathbb{1}_{\{\mathbf{x}\in e_m\}}f'\left(\frac{(\mathbf{b},\mathbb{1})}{|\mathbf{b}|}\right)\right). \tag{68}$$

Tarkastellaan odotusarvon $Ef((\mathbf{c},\mathbb{1})$ minimointitehtävää rajoituksilla $c_m \geq 0$ kaikilla m, ja $\sum_{m=1}^{M} c_m = 1$. Kuhn-Tuckerin välttämättömät ehdot määräävät, että on olemassa numerot λ, $\mu_m > 0$ siten, että jos $c_m > 0$, niin

$$\frac{\partial Ef((\mathbf{c},\mathbb{1}))}{\partial c_m} = \lambda. \tag{69}$$

Jos $c_m = 0$, niin

$$\frac{\partial Ef((\mathbf{c},\mathbb{1}))}{\partial c_m} = \lambda + \mu_m. \tag{70}$$

Nämä ehdot seuraavat epäyhtälöstä (67) ja yhtälöstä (68).
Oletetaan nyt, että algoritmi ei pääty äärellisen askelmäärän jälkeen. Askeleella k olkoon $\mathbf{c}^{(k+1)}$ päivitetty $\mathbf{c}^{(k)}$. Tällöin

$$(\mathbf{c}^{(k+1)},\mathbb{1}) - (\mathbf{c}^{(k)},\mathbb{1}) = \frac{\mathbb{1}_{\{\mathbf{x}\in e_{m^*(k)}\}} - (\mathbf{c}^{(k)},\mathbb{1})}{k+1}. \tag{71}$$

Merkitään ylläolevan yhtälön (71) oikeaa puolta $\frac{\delta_k(x_n)}{k+1}$. Taylorin lauseen perusteella

$$Ef((\mathbf{c}^{(k+1)},\mathbb{1})) - Ef((\mathbf{c}^{(k)},\mathbb{1})) = \frac{1}{k+1}E\delta_k f'((\mathbf{c}^{(k)},\mathbb{1})) + \frac{\gamma}{(k+1)^2}. \tag{72}$$

Yhtälön (72) oikean puolen ensimmäinen termi on negatiivinen kaikilla k. Koska $Ef((\mathbf{c}, \mathbb{1}))$ on rajoitettu kaikilla $\mathbf{c}$ niin

$$\sum_k \frac{1}{k+1} E\delta_k f'((\mathbf{c}^{(k)}, \mathbb{1})) < \infty. \tag{73}$$

Joten, mahdollisesti lukuunottamatta nollamittaisella jonon $(k)_1^\infty$ osajonolla,

$$E\delta_k f'((\mathbf{c}^{(k)}, \mathbb{1})) \to 0. \tag{74}$$

Tarkastellaan sellaista jonon $(k)_1^\infty$ osajonoa, jolla (74) on voimassa siten, että $m^*(k) \to m^*$, kun $\mathbf{c}^{(k)} \to \mathbf{c}$. Tästä seuraa, että yhtälö (68) on voimassa. Koska $f''(x) \geq 0$ välillä $[0, 1]$, $Ef((\mathbf{c}, \mathbb{1}))$ on konveksi, ja sellainen piste $\mathbf{c}$ joka toteuttaa Kuhn-Tuckerin ehdot on globaali minimi. Edelleen, koska yhtälön (72) oikean puolen ensimmäinen termin on negatiivinen, niin yhtälöstä (72) seuraa, että jono $Ef((\mathbf{c}^{(k)}, \mathbb{1}))$ suppenee. Siis kaikki jonon $\mathbf{c}^{(k)}$ rajapisteet ovat odotusarvon $Ef((\mathbf{c}, \mathbb{1}))$ globaaleja minimejä.
Tarkastellaan nyt tapausta $\phi \geq \phi^*$. Selvästi $f(0) \leq Ef((\mathbf{c}, \mathbb{1}))$ mille tahansa kerroinvektorille $\mathbf{c}$. Jos algoritmit pysähtyy jollakin askeleella (if there is any stopping point), koska $Ef'((\mathbf{c}, \mathbb{1})) = 0$, niin $Ef((\mathbf{c}, \mathbb{1})) = f(0)$. Muussa tapauksessa, huomattakoon että millä tahansa kerroinvektorilla $\mathbf{c}$, $E(\mathbf{c}, \mathbb{1})f'((\mathbf{c}, \mathbb{1})) \geq \phi E(\mathbf{c}, \mathbb{1})f'((\mathbf{c}, \mathbb{1}))$. Siis

$$E(\mathbb{1}_{\{\mathbf{X} \in e_{m^*}\}} - (\mathbf{c}, \mathbb{1}))f'((\mathbf{c}, \mathbb{1})) \leq (\phi^* - \phi)Ef'((\mathbf{c}, \mathbb{1})). \tag{75}$$

Jos $\phi > \phi^*$ epäyhtälön (75) oikea puoli on aidosti negatiivinen ja algoritmi ei pysähdy. Tällöin lauseen (3.10) todistuksessa käytetty perustelu antaa osajonon joka toteuttaa (74). Mille tahansa rajapisteelle $\mathbf{c}$ ja sitä vastaavalle m^*, $E(\mathbb{1}_{\{\mathbf{X} \in e_{m^*}\}} - (\mathbf{c}, \mathbb{1}))f'((\mathbf{c}, \mathbb{1})) = 0$, mistä seuraa $P((\mathbf{c}, \mathbb{1}) > \phi) = 0$. Jos $\phi = \phi^*$ ja algoritmi pysähtyy, niin $E(\mathbb{1}_{\{\mathbf{X} \in e_{m^*}\}} - (\mathbf{c}, \mathbb{1}))f'((\mathbf{c}, \mathbb{1})) = 0$, mistä seuraa $P((\mathbf{c}, \mathbb{1}) > \phi) = 0$. Jollei se pysähdy, johtopäätös on edelleen sama. Kummassakin tapauksessa saamme $Ef((\mathbf{c}, \mathbb{1})) = f(0)$. □

Yksi versio normalisoidusta algoritmista lähtee sellaisesta välillä $[-1, 1]$ määritellystä funktiosta $g(x)$, että $g'(x)$ on nolla kun $x \leq 0$ ja positiivinen, kun $x > 0$, ja g'' on jatkuva, rajoitettu ja epänegatiivinen. Määritellään $f(x) = g(x - \phi)$, missä $\phi > \phi^*$. Soveltamalla normalisoitua mukautuvaa algoritmia funktioon f saadaan seuraava tulos:

Seuraus 3.12. *Mille tahansa normalisoidun algoritmin rajapisteelle* $\mathbf{c}$, $P(\text{edge}(\mathbf{c}) > \phi) = 0$.

Todistus. $P(\text{edge}(\mathbf{c}) > \phi) = 0$ on välttämätön ja riittävä ehto sille että funktion g minimi on $g((\mathbf{c}, \mathbb{1}))$. □

Huomautus.

(i) Minimin etsintä haulla suoraa pitkin sellaiseen suuntaan johon b_{m^*} kasvaa voitaisiin tehdä myös normalisoidussa algoritmissa, ja tämä nopeuttaisi suppenemista. On epäselvää onko tämä laskennallisesti mielekästä.

(ii) $g(0) = 1$ on yläraja.

3.8.4 Universaalit mukautuvat algoritmit

Kappaleiden 3.8.1 ja 3.8.2 algoritmit sellaisen kerroinvektorin $\mathbf{c}$ löytämiseksi, että $P(\text{edge}(\mathbf{c}) > \phi) = 0$ riippuvat parametrille ϕ asetetusta arvosta. Niissä ϕ^* oletetaan tunnetuksi, valitaan kiinnitetty $\phi > \phi^*$, ja etsitään sellainen $\mathbf{c}$, että $P(\text{edge}(\mathbf{c}) > \phi) = 0$. Entä jos ei tunneta arvoa ϕ^*, onko tällöin olemassa mukautuvaa algoritmia joka suppenee kohti sellaista kerroinvektoria $\mathbf{c}$, jolla $P(\text{edge}(\mathbf{c}) > \phi) = 0$ kaikilla $\phi > \phi^*$? Seuraavaksi kuvailemme kaksi sellaista algoritmia. Molemmat ovat normalisoimattomia ja käyttävät funktiota e^x, mutta eivät käytä hakua suoraa pitkin askeleen pituuden määräämisessä.

1. universaali mukautuva algoritmi

Algoritmi 5 1. universaali mukautuva algoritmi

1: Määritellään jono askelpituuksia $(\Delta_k)_1^\infty$, jolle pätee $\Delta_k \to 0$ ja $\sum_{k=1}^\infty \Delta_k = \infty$.
2: Asetetaan $\mathbf{b}^{(0)} = 0$.
3: Kaikilla $k = 1, \ldots, K$ asetetaan
   (a) $\pi_{k-1}(n) = \prod_m e^{b_m^{(k-1)} \mathbb{1}_{(\mathbf{x}_n \in e_m)}}$, $n = 1, \ldots, N$
   (b) $Q_{k-1}(\mathbf{x}_n) = \frac{\pi_{k-1}(n)}{\sum_{n=1}^N \pi_{k-1}(n)}$, $n = 1, \ldots, N$
   (c) $m^* = \arg\min_m Q_{k-1}(e_m)$.
   (d) Päivitetään $\mathbf{b}^{(k)} \leftarrow \mathbf{b}^{(k-1)} + \Delta_k \mathbf{u}_{m^*}$.

Lause 3.13. *Mikä tahansa 1. universaalin mukautuvan algoritmin rajapiste* $\mathbf{c}$ *toteuttaa* $P(\text{edge}(\mathbf{c}) > \phi) = 0$ *kaikilla* $\phi > \phi^*$.

Todistus. Huomattakoon, että

$$P((\mathbf{b}, \mathbb{1}) > \phi|\mathbf{b}|) \leq e^{-\phi|\mathbf{b}|} E \prod_m e^{b_m \mathbb{1}_{\{\mathbf{x} \in e_m\}}}. \tag{76}$$

Kun on otettu k askelta, merkitään epäyhtälön (76) oikeaa puolta f_k ja $\epsilon_k = Q(e_{m^*})$. Tällöin

$$f_{k+1} = e^{-\phi|\mathbf{b}|} e^{-\phi\Delta_k} E e^{\Delta_k \mathbb{1}\{\mathbf{x}\in e_{m^*}\}} \prod_m e^{b_m \mathbb{1}\{\mathbf{x}\in e_m\}}, \tag{77}$$

joten

$$f_{k+1} = e^{-\phi\Delta_k}(1 + (e^{\Delta_k} - 1)\epsilon_k) f_k. \tag{78}$$

Tällöin

$$f_{K+1} = \prod_{k=1}^{K} e^{-\phi\Delta_k}(1 + (e^{\Delta_k} - 1)\epsilon_k) \tag{79}$$

ja

$$\log(f_{K+1}) = -\sum_{k=1}^{K}(\Delta_k(\phi - \epsilon_k) - O(\Delta_k^2)). \tag{80}$$

Koska $\epsilon_k \leq \phi^*$, epäyhtälöstä (80) seuraa että $f_{K+1} \to 0$ kaikilla $\phi > \phi^*$. □

1. universaali mukautuva algoritmi voi pienillä kertoimen b_{m^*} muutoksillaan olla hidas suppenemaan verrattuna normalisoimattomiin algoritmeihin jotka käyttävät hakua suoraa pitkin. Pienten vakiolisäysten Δ käyttäminen tuottaa sellaisen kerroinvektorin $\mathbf{c}$, että $P(\text{edge}(\mathbf{c}) > \phi) =$ kaikilla $\phi > \frac{\phi^*}{1-\Delta} + O(\Delta^2)$.

2. universaali mukautuva algoritmi

Algoritmi 6 2. universaali mukautuva algoritmi

1: Kiinnitetään yläraja $L < 1$ alareunalle, esim. $L = 0.9$.
2: Asetetaan $\mathbf{b}^{(0)} = 0$.
3: Kaikilla $k = 1, \ldots, K$ asetetaan

(a) $\pi_{k-1}(n) = \prod_m e^{b_m^{(k-1)} \mathbb{1}_{(\mathbf{x}_n \in e_m)}}$, $n = 1, \ldots, N$

(b) $Q_{k-1}(\mathbf{x}_n) = \frac{\pi_{k-1}(n)}{\sum_{n=1}^{N} \pi_{k-1}(n)}$, $n = 1, \ldots, N$,

(c) $m^* = \arg\min_m Q_{k-1}(e_m)$,

(d) $\epsilon_k = Q_{k-1}(e_{m^*})$

(e) $s_k = \min(\max_n(\sum_{m=1}^{M} c_m^{(k-1)} \mathbb{1}_{(\mathbf{x}_n \in e_m)}), L)$.

(f) Määritellään askelpituudeksi

$$\Delta_k = \log\left(\frac{s_k}{1 - s_k}\right) + \log\left(\frac{1 - \epsilon_k}{\epsilon_k}\right). \tag{81}$$

(g) Päivitetään $\mathbf{b}^{(k)} \leftarrow \mathbf{b}^{(k-1)} + \Delta_k \mathbf{u}_{m^*}$.

Lause 3.14. *Mikä tahansa 2. universaalin mukautuvan algoritmin rajapiste* $\mathbf{c}$ *toteuttaa* $P(\text{edge}(\mathbf{c}) > \phi) = 0$ *kaikilla* $\phi > \phi^*$.

Todistus. Askeleella k olkoon $s_k = \min(\max_n(\mathbf{c}_k, \mathbb{1}(\mathbf{x}_n)), b)$ ja $\epsilon_k = Q(e_{m^*})$. Tällöin yhtälöstä (78) seuraa

$$\frac{f_{k+1}}{f_k} = \left(\frac{1 - \epsilon_k}{1 - s_k}\right)^{1-\phi} \left(\frac{\epsilon_k}{s_k}\right)^{\phi} \tag{82}$$

Ottamalla puolittain logaritmit ja hyödyntämällä tietoa $\epsilon_k \leq \phi^*$ saadaan

$$\log\left(\frac{f_{k+1}}{f_k}\right) \leq (1-\phi)\log(1-\phi^*) + \phi\log(\phi^*) - (1-\phi)\log(1-s_k) - \phi\log(s_k). \tag{83}$$

Olkoon $\phi > \phi^*$. Funktio

$$\theta(s) = (1-\phi)\log(1-\phi^*) + \phi\log(\phi^*) - (1-\phi)\log(1-s) - \phi\log(s) \tag{84}$$

on negatiivinen kun $\phi^* < s < h(\phi)$, missä $h(\phi) > \phi$. Olkoon $\bar{s} = \limsup(s_k)$. Koska $\bar{s} \leq b$, $f_k \to 0$, kun $\phi \leq h^{-1}(b) < b$, joten $\bar{s} \leq h^{-1}(b)$. Sama päättely toistamalla saadaan johtopäätökseksi, että $\bar{s} = \phi^*$. □

Huomautus. Algoritmi 1 tuottaa sellaisen kerroinvektorin $\mathbf{c}$, että $P(\text{edge}(\mathbf{c}) > \phi) = 0$ kaikilla

$$\phi > \frac{\log(2) + \log(1 - \phi^*)}{-\log(\phi^*) + \log(1 - \phi^*)}. \tag{85}$$

Esimerkiksi, jos $\phi^* = 0.25$, alaraja epäyhtälössä (85) on 0.37.

3.9 Vahvistetut puut

Regressio- ja luokittelupuut jakavat selittäjämuuttujien avaruuden pistevieraisiin alueisiin R_j, $j = 1, \ldots, J$. Jokaista tällaista aluetta R_j kohti valitaan γ_j, ja asetetaan ennustussäännöksi

$$x \in R_j \implies f(x) = \gamma_j. \tag{86}$$

Siis puu voidaan formaalisti esittää

$$T(x; \Theta) = \sum_{j=1}^{J} \gamma_j \mathbb{1}_{(x \in R_j)}, \tag{87}$$

parametreilla $\Theta = \{R_j, \gamma_j\}_1^J$. J käsitetään tavallisesti metaparametrina. Parametrit saadaan minimoimalla empiirinen riski

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{j=1}^{J} \sum_{x_i \in R_j} L(y_i, \gamma_j). \tag{88}$$

Tämä on valtava kombinatorinen optimointiongelma, joten tässä kohtaa yleensä tyydytään approksimatiivisiin suboptimaalisiin ratkaisuihin. Optimointiongelma kannattaa jakaa kahteen osaan:

(a) **Vakion γ_j löytäminen annetulla $\mathbf{R}_j$**: Annetulla R_j vakion γ_j estimoiminen on usein triviaalia: usein on $\hat{\gamma}_j = \bar{y}_j$, missä $\bar{y}_j$ on alueelle R_j osuvien aritmeettinen keskiarvo. For mis-classification loss, $\hat{\gamma}_j$ is the modal class of observations falling in region R_j.

(b) **Alueen $\mathbf{R}_j$ löytäminen**: Tämä on näistä kahdesta vaikeampi vaihe, jolle on löydetty likimääräisiä ratkaisuja. Huomautettakoon myös, että alueen R_j löytäminen sisältää myös vakion γ_j estimoimisen. Tyypillinen strategia on käyttää ahnetta ylhäältä alas -rekursiivista ositusalgoritmia alueen R_j löytämiseksi. Lisäksi joskus on tarpeen approksimoida minimointiongelmaa (88) sileämmällä ja kätevämmällä kriteerillä alueen R_j optimoimiseksi:

$$\tilde{\Theta} = \arg\min_{\Theta} \sum_{i=1}^{N} \tilde{L}(y_i, T(x_i, \Theta)). \tag{89}$$

Nyt asettamalla $\hat{R}_j = \tilde{R}_j$ vakio γ_j voidaan estimoida tarkemmin käyttäen alkuperäistä kriteeriä.

Aiemmin kappaleessa 2.2 kuvailimme tällaisen strategian luokittelupuille. Gini-indeksi korvasi luokitteluvirheen puun kasvattamisessa (alueen R_j löytämisessä).

Vahvistettu puumalli on kaavassa (87) esitettyjen puiden summa:

$$f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m) \tag{90}$$

eteenpäin vaihettaisesti johdettuna (Algoritmi 2). Joka askeleella eteenpäin vaiheittaisessa menettelyssä ratkaistaan

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m)) \tag{91}$$

seuraavan puun parametrien (eli alueiden ja sovitteiden) $\Theta_m = \{R_{jm}, \gamma_{jm}\}_1^{J_m}$ suhteen, vallitsevan mallin $f_{m-1}(x)$ pohjalta. Alueiden R_{jm} ollessa määrättyjä, niitä vastaavien optimaalisten vakioiden γ_{jm} löytäminen on tyypillisesti suoraviivaista:

$$\hat{\gamma}_{jm} = \arg\min_{\gamma_{jm}} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma_{jm}). \tag{92}$$

Alueiden löytäminen on vaikeaa, vieläkin vaikeampaa kuin yhden puun tapauksessa. Ongelma helpottuu muutamassa erikoistapauksessa.

Neliövirheelle ongelman (91) ratkaiseminen ei ole vaikeampaa kuin yksittäisen puun tapauksessa. Se on yksinkertaisesti se regressiopuu joka parhaiten ennustaa vallitsevan mallin residuaaleja $y_i - f_{m-1}(x_i)$, ja $\hat{\gamma}_{jm}$ on näiden residuaalien keskiarvo vastaavassa alueessa.

Kahden luokan luokittelun ja eksponentiaalisen tappion tapauksessa tästä vaiheittaisesta lähestymistavasta seuraa AdaBoost-menetelmä luokittelupuiden vahvistamiseksi (Algoritmi 1). Erityisesti, jos puissa $T(x; \Theta_m)$ rajoitutaan skaalattuihin luokittelupuihin, niin kappaleen 3.4 perusteella optimoin-

tiongelman (91) ratkaisu on sellainen puu joka minimoi painotetun virhesumman $\sum_{i=1}^{N} w_i^{(m)} \mathbb{1}_{(y_i \neq T(x_i;\Theta_m))}$, painoinaan $w_i^{(m)} = e^{-y_i f_{m-1}(x_i)}$. Skaalatulla luokittelupuulla tarkoitamme puuta $\beta_m T(x;\Theta_m)$ rajoituksella $\gamma_{jm} \in \{-1, 1\}$. Ilmankin tätä rajoitusta ongelma (91) pelkistyy eksponentiaaliseksi tappioksi painotetuilla eksponentiaalisilla kriteereillä:

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} w_i^{(m)} \exp\left[-y_i T(x_i;\Theta_m)\right]. \tag{93}$$

On suoraviivaista implementoida ahne rekursiivinen ositusalgoritmi käyttäen tätä painotettua eksponentiaalista tappiota jakokriteerinä. Huomautettakoon, että annetulla R_{jm} ongelman (92) ratkaisu on painotettu log-odds kussakin alueessa:

$$\hat{\gamma}_{jm} = \log \frac{\sum_{x_i \in R_{jm}} w_i^{(m)} \mathbb{1}_{(y_i=1)}}{\sum_{x_i \in R_{jm}} w_i^{(m)} \mathbb{1}_{(y_i=-1)}}. \tag{94}$$

Tämä vaatii erityisen puunkasvatusalgoritmin; käytännössä suosimme edempänä esiteltyä likimääräistä approksimaatiota, joka käyttää painotettua pienimmän neliösumman regressiopuuta.

Absoluuttisen virheen tai Huberin virheen tapaisten virhe- (tai tappio-) kriteerien käyttö neliövirheen asemesta regressiossa, ja devianssin käyttö eksponentiaalisen virheen asemesta luokittelussa tuo vahvistuspuihin robustisuutta. Valitettavasti, toisin kuin vähemmän robustit vastineensa, nämä robustit kriteerit eivät johda yksinkertaisiin nopeisiin vahvistusalgoritmeihin.

Yleisemmillä virhekriteereillä ongelman (92) ratkaisu annetulla R_{jm} on tyypillisesti suoraviivainen, koska se on yksinkertainen lokaali estimaatti. Absoluuttiselle virheelle (L1-virheelle) se on yksinkertaisesti residuaalien mediaani kullakin alueella. Muilla kriteereille on olemassa nopeita iteratiivisia algoritmeja, ja usein niiden nopeammat yhden askeleen approksimaatiot ovat riittäviä. Vaikeutena tässä on puun johtaminen. Ongelman (91) ratkaisemiseksi ei ole olemassa yksinkertaisia nopeita algoritmeja näillä yleisemmillä tappiokriteereillä, jolloin approksimaatiot kuten (89) ovat hyödyllisiä.

3.10 Numeerinen optimointi

Nopeat likimääräiset algoritmit tehtävän (91) ratkaisemiseksi millä tahansa differentioituvalla tappiokriteerillä voidaan analogisesti johtaa numeeriseen

optimointiin. Tappio käytettäessä funktioita $f(x)$ vasteen y ennustamisessa on

$$L(f) = \sum_{i=1}^{N} L(y_i, f(x_i)). \tag{95}$$

Tavoite on minimoida $L(f)$ funktion f suhteen, sillä rajoituksella, että f on puiden summa kuten kaavassa (90). Jos tämä rajoitus jätetään huomiotta, (95) voidaan ajatella numeerisena optimointitehtävänä

$$\hat{\mathbf{f}} = \arg\min_{\mathbf{f}} L(\mathbf{f}), \tag{96}$$

missä "parametrit" $\mathbf{f} \in \mathbb{R}^N$ ovat likimääräisen funktion arvoja $f(x_i)$ kussakin datapisteessä x_i:

$$\mathbf{f} = \{f(x_1), f(x_2), \ldots, f(x_N)\}. \tag{97}$$

Numeeriset optimointimenetelmät ratkaisevat tehtävän (96) komponenttivektorien summana

$$\mathbf{f}_M = \sum_{m=0}^{M} \mathbf{h}_m, \quad \mathbf{h}_m \in \mathbb{R}^N, \tag{98}$$

missä $\mathbf{f}_0 = \mathbf{h}_0$ on alkuarvaus, ja jokainen uusi $\mathbf{f}_m$ johdetaan edellisen parametrivektorin $\mathbf{f}_{m-1}$ perusteella, joka on aiemmin johdettujen päivitysten summa. Numeeriset optimointimenetelmät poikkeavat toisistaan siinä, miten kukin lisäysvektori $\mathbf{h}_m$ (askel) lasketaan.

3.10.1 Jyrkimmän laskun menetelmä

Jyrkimmän laskun menetelmässä $\mathbf{h}_m = -\rho_m \mathbf{g}_m$, missä ρ_m on skalaari ja $\mathbf{g}_m \in \mathbb{R}^N$ on funktion $L(\mathbf{f})$ gradientti evaluoituna kohdassa $\mathbf{f} = \mathbf{f}_{m-1}$. Gradientin $\mathbf{g}_m$ komponentit (koordinaatit) ovat

$$g_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)}. \tag{99}$$

Askelpituus ρ_m on

$$\rho_m = \arg\min_{\rho} L(\mathbf{f}_{m-1} - \rho \mathbf{g}_m). \tag{100}$$

Iteraatiota päivitetään nyt

$$\mathbf{f}_m = \mathbf{f}_{m-1} - \rho \mathbf{g}_m, \tag{101}$$

ja tällä tavoin prosessia jatketaan. Jyrkimmän laskun menetelmä on *ahne* strategia, sillä $-\mathbf{g}_m$ on avaruuden $\mathbb{R}^N$ se suunta johon $L(\mathbf{f})$ vähenee nopeiten kohdassa $\mathbf{f} = \mathbf{f}_{m-1}$.

3.10.2 Gradienttivahvistaminen (Gradient boosting)

Eteenpäin vaiheittainen vahvistaminen (Algoritmi 2) on myös ahne strategia. Joka askeleella ratkaisupuu on se puu joka maksimaalisesti vähentää arvoa (91) ehdolla vallitseva malli f_{m-1} ja sen sovitukset $f_{m-1}(x_i)$. Puuennusteet $T(x_i, \mathbf{\Theta}_m)$ ovat analogisia negatiivisen gradientin komponenttien kanssa (99). Pääasiallinen ero niiden välillä on se, että puun komponentit $\mathbf{t}_m = (T(x_1; \mathbf{\Theta}_m), \ldots, T(x_N; \mathbf{\Theta}_m))$ eivät ole riippumattomia. Ne ovat rajoitetut sellaisen päätöspuun ennustuksiksi, jossa on J_m kpl lehtiä, kun taas negatiivinen gradientti on rajoittamaton maksimaalisen vähenemisen suunta.

Tehtävän (92) ratkaisu vaiheittaisessa menettelyssä on analoginen jyrkimmän laskun menetelmän askelpituuden haun kanssa. Ero on siinä, että (92) suorittaa erillisen haun niille vektorin $\mathbf{t}_m$ komponenteille, jotka vastaavat erillistä puun lehtisolmuja $\{T(x_i; \mathbf{\Theta}_m)\}_{x_i \in R_{jm}}$.

Jos opetusdatan tappion (95) minimointi olisi ainoa tavoite, jyrkimmän laskun menetelmä olisi suositeltu strategia. Gradientin (99) laskeminen on triviaalia mille tahansa differentioituvalle tappiofunktiolle $L(y, f(x))$, kun taas tehtävän (91) ratkaiseminen on vaikeaa kappaleessä (3.6) käsitellyillä robusteilla kriteereillä. Valitettavasti gradientti (99) on määritelty vain vain opetusdatapisteissä x_i ja lopullisena tavoitteena on yleistää $f_M(x)$ datalle joka ei kuulu opetusdatan joukkoon.

Yksi mahdollinen ratkaisu tähän ongelmaan on indusoida iteraatiolla m puu $T(x; \mathbf{\Theta}_m)$ jonka ennusteet $\mathbf{t}_m$ ovat niin lähellä negatiivista gradienttia kuin mahdollista. Käyttäen neliöllistä virhettä läheisyysmittana, tämä johtaa minimointitehtävään

$$\tilde{\mathbf{\Theta}}_m = \arg\min_{\Theta} \sum_{i=1}^{N} (-g_{im} - T(x_i; \mathbf{\Theta}))^2. \tag{102}$$

Sovitetaan siis puu T negatiivisen gradientin arvoihin (99) pienimmän neliösumman menetelmällä. Kuten kappaleessa (3.9) todettiin, pns-päätöspuun johtamiselle on olemassa nopeita algoritmeja. Vaikka tehtävän (102) ratkaisualueet $\tilde{R}_{jm}$ eivät ole identtisiä tehtävän (91) ratkaisualueiden R_{jm} kanssa,

Asetelma	Tappiofunktio	$-\partial L(y_i, f(x_i))/\partial f(x_i)$
Regressio	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$
Regressio	$\lvert y_i - f(x_i)\rvert$	$\text{sign}[y_i - f(x_i)]$
Regressio	Huber	$y_i - f(x_i)$ kun $\lvert y_i - f(x_i)\rvert \leq \delta_m$
		$\delta_m \text{sign}[y_i - f(x_i)]$ kun $\lvert y_i - f(x_i)\rvert > \delta_m$,
		$\delta_m = \alpha-\text{kvantiili}\{\lvert y_i - f(x_i)\rvert\}$
Luokittelu	Devianssi	k:s komponentti: $\mathbb{1}_{(y_i=\mathcal{G}_k)} - p_k(x_i)$

Taulukko 1: Tappiofunktioiden gradientteja

ne ovat yleensä tarpeeksi lähellä ajaakseen saman asian. Joka tapauksessa eteenpäin vaiheittainen vahvistusproseduuri ja ylhäältä alas päätöspuun johtaminen ovat sinällään approksimaatiomenetelmiä. Puun (102) konstruoinnin jälkeen vastaavat vakiot kussakin alueessa ovat (92).

Taulukosta 1 löytyy yleisimpien tappiofunktioiden gradientteja. Neliövirhetappiolle negatiivinen gradientti on vain tavallinen residuaali $-g_{im} = y_i - f_{m-1}(x_i)$, joten (102) vastaa tavallisen pns-menetelmän vahvistamista. Absoluuttiselle virheelle (toinen rivi) negatiivinen gradientti on residuaalin *merkki*, niin että joka iteraatiolla (102) sovittaa puun sen hetkisten residuaalien merkkiin pns-menetelmällä. Huberin M-regressiolle negatiivinen gradientti on kahden edellisen tapauksen kompromissi (ks.taulukko).

Luokittelussa tappiofunktio on multinomidevianssi (38), ja joka iteraatiolla konstruoidaan K kpl pns-puita. Jokainen puu T_{km} sovitetaan sitä vastaavaan negatiiviseen gradienttivektoriin g_{km},

$$\begin{aligned} -g_{ikm} &= \frac{\partial L(y_i, f_{1m}(x_i), \ldots, f_{Km}(x_i)}{\partial f_{km}(x_i)} \\ &= \mathbb{1}_{(y_i=\mathcal{G}_k)} - p_k(x_i), \end{aligned} \tag{103}$$

missä kertoimet $p_k(x)$ löytyvät kaavasta (37). Vaikka joka iteraatiolla rakennetaankin K kpl erillisiä puita, ne liittyvät toisiinsa kaavan (37) kautta.

3.10.3 MART

Algoritmi 7 on yleinen gradienttipuun vahvistusalgoritmi regressiolle. Vaihtelemalla tappiofunktiota $L(y, f(x))$ saadaan erikoistapauksia. Tähän lähestymistapaan viitataan kirjainlyhenteellä MART (multiple additive regression trees). Algoritmin ensimmäinen rivi alustaa optimaalisen vakiomallin, joka

Algoritmi 7 Gradienttipuuvahvistaminen monen muuttujan additiiviselle regressiopuulle

1: Alusta $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2: Jokaiselle $m = 1, \ldots, M$:

(a) Jokaiselle $i = 1, 2, \ldots, N$ laske

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$

(b) Sovita regressiopuu arvoihin r_{im}. Saadaan lehtialueet (terminal regions) $R_{jm},\ j = 1, 2, \ldots, J_m$.

(c) Jokaiselle $j = 1, 2, \ldots, J_m$ laske

$$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma).$$

(d) Päivitä $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbb{1}_{(x \in R_{jm})}$.

3: Palauta $\hat{f}(x) = f_M(x)$.

koostuu ainoastaan yhdestä lehtisolmusta. Negatiivisen gradientin komponentteja, jotka lasketaan rivillä $2(a)$, kutsutaan yleistetyiksi tai *pseudo*residuaaleiksi, r. Tavallisesti käytettyjen tappiofunktioiden gradientteja on listattu Taulukossa 1.

Algoritmi luokittelulle on samankaltainen. Rivit $2(a) - (d)$ toistetaan K kertaa joka iteraatiolla, kerran jokaiselle luokalle kaavan

$$\begin{aligned} -g_{ikm} &= \left[\frac{\partial L(y_i, f_1(x_i), \ldots, f_K(x_i))}{\partial f_k(x_i)}\right]_{\mathbf{f}(x_i)=\mathbf{f}_{m-1}(x_i)} \\ &= \mathbb{1}_{(y_i = \mathcal{G}_k)} - p_k(x_i), \end{aligned} \tag{104}$$

missä

$$p_k(x) = \frac{e^{f_k(x)}}{\sum_{\ell=1}^{K} e^{f_\ell(x)}}, \tag{105}$$

mukaan. Lopputuloksena rivillä 3 saadaan K kpl erilaisia puulaajennuksia $f_{kM}(x)$, $k = 1, 2, \ldots, K$. Nämä tuottavat todennäköisyyksiä (37) tai tekevät luokittelua (36).

MART-proseduurin parametrien säätöparametrit ovat iteraatioiden lukumäärä M, ja puiden J_m, $m = 1, 2, \ldots, M$, koot.

3.11 Oikean kokoinen puu vahvistusta käytettäessä

Historiallisesti vahvistuksen on ajateltu olevan tapa yhdistää malleja (tässä tapauksessa puita). Puunrakennusalgoritmi ajatellaan primitiiviseksi tavaksi tuottaa heikkoja malleja, jotka yhdistetään vahvistamalla. Tässä tilanteessa optimaalinen puun koko estimoidaan tavanomaisella tavalla erikseen jokaiselle puulle rakennusvaiheessa kuten kappaleessa 2.

3.12 Regularisaatio

Puiden koon J lisäksi gradienttivahvistamisen toinen metaparametri on vahvistusiteraatioiden lukumäärä M. Jokainen iteraatio tavallisesti vähentää opetusriskiä $L(f_M)$, niin että kyllin isolla M tämä riski saadaan mielivaltaisen pieneksi. Liian tarkka sovitus opetusdataan voi kuitenkin johtaa ylisovittumiseen, joka huonontaa ennustusten riskiä. On siis olemassa tilanteesta riippuva optimaalinen vahvistusiteraatioiden lukumäärä M^*. Kätevä tapa estimoida optimaalista parametrin arvoa M^* on seurata ennustusriskiä parametrin M funktiona validointiotoksella. Optimaalisen arvon M^* estimaatiksi valitaan tällöin ennustusriskin minimoiva parametrin M arvo.

3.12.1 Kutistaminen

Vahvistusiteraatioiden lukumäärän säätäminen ei ole ainoa mahdollinen regularisaatiostrategia. Yksi mahdollinen reguralisaatiomenetelmä on *kutistaminen*. Yksinkertaisin tapa soveltaa kutistamista vahvistamisen yhteydessä on skaalata kutakin malliin lisättävää puuta kertoimella $0 < \nu < 1$. Siis Algoritmin 7 rivi 2(d) korvataan sijoituksella

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot \sum_{j=1}^{J} \gamma_{jm} \mathbb{1}_{(\mathbf{x} \in R_{jm})}, \tag{106}$$

missä siis heikko oppija $h(\mathbf{x}, \mathbf{a}_m)$ on regressiopuu $\sum_{j=1}^{J} \gamma_{jm} \mathbb{1}_{(\mathbf{x} \in R_{jm})}$.

Parametri ν kontrolloi vahvistamisen oppimisnopeutta. Pienemmät parametrin ν arvot johtavat suurempaan opetusvirheeseen samalla vahvistusiteraatioiden M lukumäärällä. Siis sekä ν, että M hallitsevat ennustusriskiä opetusdatalla. Nämä eivät kuitenkaan ole toisistaan riippumattomia. Pienemmät parametrin ν arvot johtavat suurempaan parametrin M arvoihin opeturvirheen pysyessä vakiona.

Empiirisesti on havaittu [7], että pienemmät parametrin ν arvot, jotka vaativat vastaavasti suurempia parametrin M arvoja, johtavat pienempään testivirheeseen. Itse asiassa, paras strategia näyttäisi olevan parametrin ν

asettaminen hyvin pieneksi ($\nu < 0.1$), pysäyttämällä vahvistusproseduuri, kun validointivirhe on suurempi verrattuna edelliseen iteraatioon, ja valitsemalla tämä edellisen iteraation M optimaaliseksi vahvistusiteraatioiden lukumääräksi. Tämä strategia tuottaa yllättävän suuria parannuksia (verrattuna tilanteeseen $\nu = 1$) regressiotilanteessa ja todennäköisyyksien estimoinnissa. Vastaavat parannukset luokitteluvirheen riskiin ovat pienemmät, mutta silti huomattavat. Hinta näistä parannuksista on laskennallinen: pienemmät parametrin ν arvot johtavat suurempiin parametrin M arvoihin, ja laskennallinen hinta riippuu jälkimmäisestä. Suuri määrä iteraatioita on kuitenkin toteutettavissa jopa hyvin suurilla aineistoilla. Tämä johtuu osittain siitä, että yksittäiset malliin lisättävät puut ovat pieniä, eikä karsimista tarvitse tehdä.

3.12.2 Osanäytteidenotto

Stokastisessa gradienttivahvistamisessa otamme joka iteraatiolla näytteen (ilman takaisinpanoa) joka on kooltaan murto-osa η koko opetusaineistosta, ja kasvatamme tämän perusteella seuraavan puun. Muuten algoritmi on identtinen aiemmin esitetyn kanssa. Tyypillinen arvo parametrille η on $\frac{1}{2}$, mutta suurille havaintoaineistoille se voi olla huomattavasti pienempi.

Tämä strategia ei vain vähennä laskenta-aikaa, vaan monissa tapauksissa tuottaa myös tarkempia malleja. Huonona puolena on, että parametreja on nyt neljä kappaletta: J, M, ν ja η.

3.13 Tulkintaa

Siinä missä yksittäiset päätöspuut ovat hyvin tulkittavissa, näiden lineaarikombinaatioiden tulkinta on mutkikkaampaa.

3.13.1 Selittäjämuuttujien suhteellinen merkitsevyys

Datanlouhintasovelluksissa selittävämuuttujat ovat harvoin yhtä merkittäviä. Usein vain muutamalla on huomattava vaikutus vasteeseen.

Yksittäiselle päätöspuulle T määrittelemme

$$\mathcal{I}_\ell^2(T) = \sum_{t=1}^{J-1} \hat{i}_t^2 \mathbb{1}_{(v(t)=\ell)} \tag{107}$$

kunkin selittäjämuuttujan X_ℓ merkityksellisyysmitaksi. Summa lasketaan puun sisäsolmujen ($J-1$ kpl) yli. Kussakin tällaisessa solmussa t, solmu

jaetaan kahteen alueeseen jonkin selittäjämuuttujan $X_{v(t)}$ suhteen, ja molemmissa alueissa vasteen arvoihin sovitetaan vakio. Valittu selittäjämuuttuja on sellainen, joka tuottaa maksimaalisen estimoidun vähennyksen neliölliselle virheelle, tilanteessa jossa alueisiin sovitetaan vakio. Muuttujan X_ℓ suhteellinen neliöllinen merkitsevyys on tällaisten neliöllisten vähennysten summa yli sellaisten sisäsolmujen, joissa se valittiin muuttujaksi, jonka suhteen ko. solmun havainnot jaetaan kahteen osaan.

Tämä merkitsevyysmitta voidaan yleistää additiivisiin puukehitelmiin (90): Lasketaan keskiarvo kaikkien puiden yli

$$\mathcal{I}_\ell^2 = \frac{1}{M}\sum_{m=1}^{M}\mathcal{I}_\ell^2(T_m). \tag{108}$$

Keskiarvoistuksen vakauttavan vaikutuksen takia tämä mitta on luotettavampi kuin aiempi, yksittäiselle puulle lasketta merkityksellisyysmitta (107). Ilmiö, jossa selittäjä peittää toisen selittäjän merkityksellisyyden, jonka kanssa se on vahvasti korreloitunut, muodostuu kutistusta käytettäessä paljon vähemmän ongelmaksi. Kaavat (107) ja (108) viittaavat *neliölliseen* merkityksellisyyteen; varsinaiset merkityksellisyydet ovat näiden neliöjuuret. Koska nämä mitat ovat suhteellisia, ne tavallisesti skaalataan siten että suurimman arvon saavan selittäjän merkityksellisyysmitta on 100.

3.13.2 Osittaisriippuvuuskuviot

Kun merkityksellisimmät muuttujat on tunnistettu, huomio suuntautuu luontaisesti siihen, miten ne vaikuttavat aproksimaatioon $f(\mathbf{X})$. Graafiset esitykset aproksimaatiosta $f(\mathbf{X})$ argumettiensa funktiona antavat yleensä kattavan yhteenvedon tästä riippuvuussuhteesta.

Valitettavasti tällaiset visualisaatiot ovat rajoitettuja mataliin dimensioihin.

Tarkastellaan osavektoria $\mathbf{X}_\mathcal{S}$, joka sisältää $\ell < p$ selittäjää koko selittäjävektorista $\mathbf{X}^T = (\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_p)$, indeksijoukkonaan $\mathcal{S} \subset \{1, 2, \ldots, p\}$. Olkoon joukko $\mathcal{C}$ indeksijoukon $\mathcal{S}$ komplementti, siis sellainen että $\mathcal{S} \cup \mathcal{C} = \{1, 2, \ldots, p\}$. Funktio $f(\mathbf{X})$ riippuu yleisessä tapauksessa kaikista selittäjämuuttujista: $f(\mathbf{X}) = f(\mathbf{X}_\mathcal{S}, \mathbf{X}_\mathcal{C})$. Yksi tapa määritellä funktion $f(\mathbf{X})$ keskimääräinen tai *osittainen* riippuvuus selittäjistä $\mathbf{X}_\mathcal{S}$ on

$$f_\mathcal{S}(\mathbf{X}_\mathcal{S}) = \mathrm{E}_{\mathbf{X}_\mathcal{C}} f(\mathbf{X}_\mathcal{S}, \mathbf{X}_\mathcal{C}). \tag{109}$$

Tämä on funktion f marginaalinen keskiarvo, ja se voi toimia hyödyllisenä kuvauksena valitun osajoukon vaikutuksesta funktioon $f(\mathbf{X})$, kun muuttujat $\mathbf{X}_\mathcal{S}$ eivät ole vahvassa interaktiossa muuttujien $\mathbf{X}_\mathcal{C}$ kanssa.

Osittaisriippuvuusfunktioiden avulla voidaan tulkita minkä tahansa musta laatikko -metodien tuloksia. Käytännössä niitä voidaan estimoida kaavalla

$$\bar{f}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}}) = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{X}_{\mathcal{S}}, x_{i\mathcal{C}}), \tag{110}$$

missä $\{x_{1\mathcal{C}}, x_{2\mathcal{C}}, \ldots, x_{N\mathcal{C}}\}$ ovat muuttujien $\mathbf{X}_{\mathcal{C}}$ opetusdatassa esiintyvät arvot. Tämä vaatii että koko aineisto käydään läpi erikseen jokaista sellaista selittäjäjoukkoa $\mathbf{X}_{\mathcal{S}}$ kohti, jolle halutaan laskea $\bar{f}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}})$. Tämä voi olla laskennallisesti raskasta jo kohtaisen kokoisilla aineistoilla. Päätöspuiden tapauksessa $\bar{f}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}})$ voidaan kuitenkin laskea nopeasti jo pelkästä puusta.

On tärkeää huomata, että kaavan (109) osittaisriippuvuusfunktio kuvaa selittäjien $\mathbf{X}_{\mathcal{S}}$ vaikutusta funktioon $f(\mathbf{X})$ *sen jälkeen* kun muiden muuttujien $\mathbf{X}_{\mathcal{C}}$ keskimääräiset vaikutukset on jo otettu huomioon. Kyseessä *ei* siis ole selittäjien $\mathbf{X}_{\mathcal{S}}$ vaikutus funktioon $f(\mathbf{X})$ siinä tilanteessa, että muiden muuttujien $\mathbf{X}_{\mathcal{C}}$ vaikutukset jätetään huomiotta. Jälkimmäinen saadaan ehdollisena odotusarvona

$$\tilde{f}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}}) = \mathrm{E}(f(\mathbf{X}_{\mathcal{S}}, \mathbf{X}_{\mathcal{C}})|\mathbf{X}_{\mathcal{S}}), \tag{111}$$

ja se on paras pienimmän neliösumman aproksimaatio funktiolle $f(\mathbf{X})$ selittäjien $\mathbf{X}_{\mathcal{S}}$ (ja ainoastaan niiden) funktiona. Suureet $\tilde{f}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}})$ ja $\bar{f}_{\mathcal{S}}(\mathbf{X}_{\mathcal{S}})$ ovat samat ainoastaan siinä epätodennäköisessä tapauksessa, jossa $\mathbf{X}_{\mathcal{S}}$ ja $\mathbf{X}_{\mathcal{C}}$ ovat riippumattomia toisistaan. Esimerkiksi jos valittujen selittäjäjoukkojen vaikutus sattuu olemaan puhtaasti additiivinen

$$f(\mathbf{X}) = h_1(\mathbf{X}_{\mathcal{S}}) + h_2(\mathbf{X}_{\mathcal{C}}). \tag{112}$$

Tällöin (109) tuottaa additiivista vakiota vaille termin $h_1(\mathbf{X}_{\mathcal{S}})$. Jos vaikutus on puhtaasti multiplikatiivinen

$$f(\mathbf{X}) = h_1(\mathbf{X}_{\mathcal{S}}) \cdot h_2(\mathbf{X}_{\mathcal{C}}), \tag{113}$$

(109) tuottaa vakiokerrointa vaille termin $h_1(\mathbf{X}_{\mathcal{S}})$. Toisaalta (111) ei tuota termiä $h_1(\mathbf{X}_{\mathcal{S}})$ kummassakaan tapauksessa. Itse asiassa (111) voi tuottaa vahvoja riippuvuuksia sellaisista selittäjäjoukoista, joista $f(\mathbf{X})$ ei riipu lainkaan.

Vahvistuspuuaproksimaation (90) osittaisriippuvuuskuvien tarkasteleminen valituilla selittäjien osajoukoilla voi auttaa tarjoamaan kvalitatiivista tulkintaa sen ominaisuuksista.

4 Gradienttivahvistaminen simuloidulla havaintoaineistolla

Ensiksi hieman taustaa. Käyttämäni data on poimittu simulaattorista, joka simuloi tietoliikennettä mobiiliaseman (UE, käytännössä useimmiten matkapuhelin) ja tukiaseman (BS tai BTS) välillä. Todellisuudessa UE:n ja BS:n välillä kulkeva signaali on tietysti reaalinen, mutta laskennallisista syistä on kätevää ajatella signaalit kompleksisina. Käyttämäni data on PUCCH-kanavan (physical uplink control channel) dataa. Tässä tarkemmin yksityiskohtiin menemättä sanottakoon että kyseessä on siis eräänlainen mobiiliasemasta tukiasemaan (uplink) kulkeva signaali. Signaalilla on simulaattorissa kaksi eri esitysmuotoa: FLP- ja FXP-esitys. FLP (floating point) tarkoittaa tietokoneen liukulukuesitystä. FXP (fixed point) tarkoittaa sellaista FLP-muodon approksimaatiota, jossa desimaalien lukumäärä ja desimaalipisteen paikka on ennalta kiinnitetty. FXP-esityksen käyttö johtuu laitteistorajoituksista ja -vaatimuksista.

Kompleksisen FLP- ja FXP-signaalien välistä etäisyyttä kompleksitasossa kutsutaan error vector magnitudeksi (EVM). Tässä yhteydessä luonnollisesti olisi suotavaa että EVM olisi mahdollisimman pieni. Olen työssäni mallintanut EVM:n käyttäytymistä gradienttivahvistetuilla regressiopuilla neljän muun muuttujan suhteen, neljässä eri kanavassa. Kanavat ovat nimeltään AWGN (additive white gaussian noise), EPA (extended pedestrian model A), TDLC ja TDLA. Neljä muuttujaa ovat signaalikohinasuhde (SNR) taajuussiirtymä (frequency offset, FOS), aikasiirtymä (time offset, TOS) ja mobiililaitteen nopeus (VEL). Muuttujien vaihteluvälit on valittu niin, että ne kattavat tukiaseman tyypillisen toimintaympäristön.

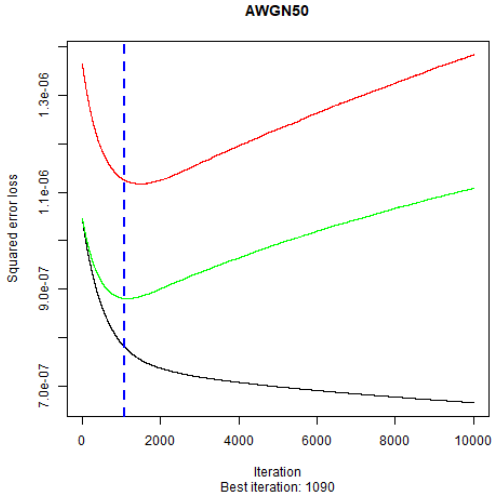
Yhtä havaintoa varten on jokainen näistä muuttujista poimittu satunnaisesti tasaisesta jakaumasta vaihteluväliltään, ajettu simulaatio ja poimittu vastaava EVM-arvo. Kutakin kanavaa kohti on kerätty oma, tuhannen havainnon kokoinen aineistonsa.

Malli on sovitettu R-ohjelmointikielen gbm-kirjastolla.

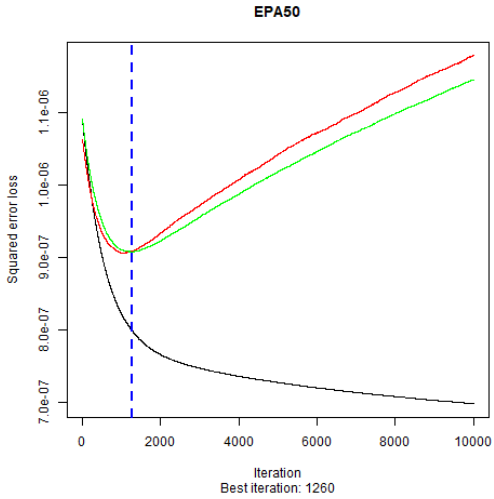
4.1 Tuloksia

Seuraavien kuvien malleissa on iteroitu 10000 puuta vaimennuskertoimena (shrink) 0.001. Puun maksimisyvyys on ollut 4 (eikä se tämän syvempi tässä tilanteessa voisi ollakaan). Kullakin kanavalla on omat havaintoainestonsa, jossa kussakin on 1000 havaintoa. Kunkin kanavan kohdalla on sovitettu oma malli siten, että puolet havainnoista on valittu satunnaisesti opetusaineistok-

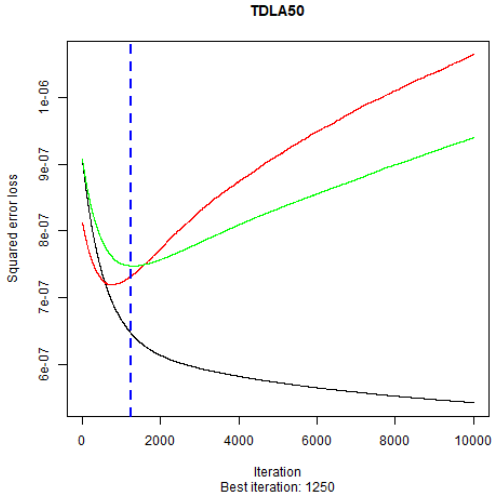
si, ja toinen puoli testiaineistoksi. Optimaalisen iteraatiomäärän kriteerinä on käytetty 5-pois-ristiinvalidoinnin virhettä. Alla olevissa kuvissa musta käyrä on virhe opetusaineistossa, punainen on validointivirhe, eli virhe testiaineistossa, ja vihreä, 5-pois-ristiinvalidoinnin virhe.



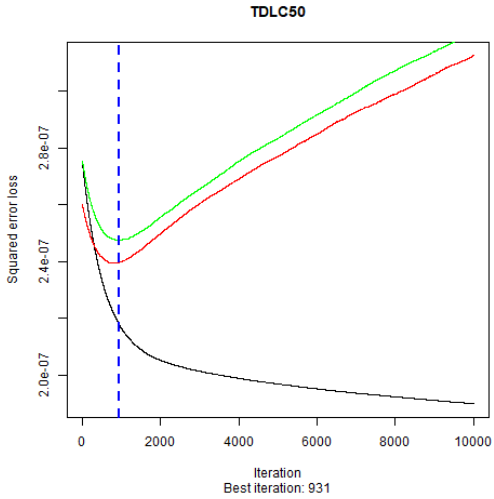
Kuva 1: Suppeneminen AWGN



Kuva 2: Suppeneminen EPA

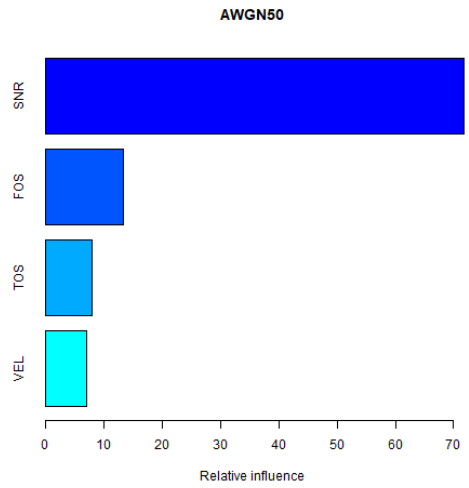


Kuva 3: Suppeneminen TDLA

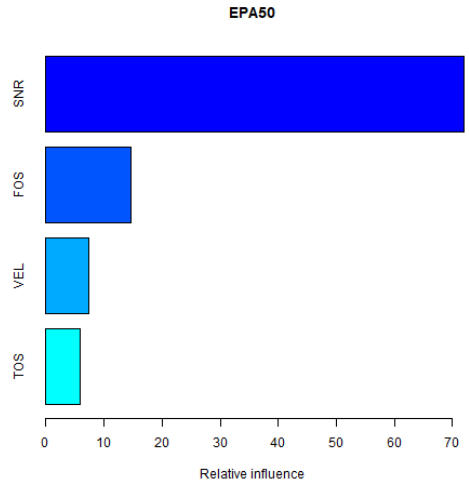


Kuva 4: Suppeneminen TDLC

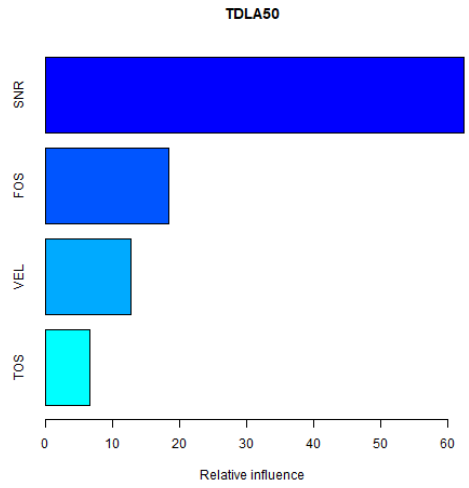
Seuraavissa kuvissa muuttujien painoarvot. Kullekin muuttujalle tämä luku kertoo kuinka paljon avaruuden jakaminen jostakin pisteestä tämän muuttujan suhteen on yhteensä pienentänyt tappiofunktion arvoa. Tappiofunktiona tässä tapauksessa on neliöllisten virheiden summa: $L(Y, f(X)) = \sum_{i=1}^{N}(Y_i - f(\hat{X}_i))^2$.



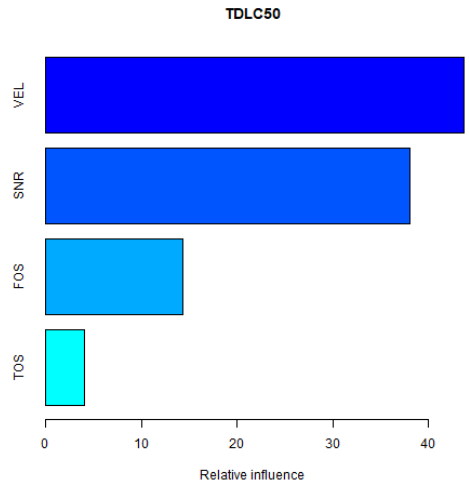
Kuva 5: Selittäjien suhteellinen merkitsevyys AWGN



Kuva 6: Selittäjien suhteellinen merkitsevyys EPA



Kuva 7: Selittäjien suhteellinen merkitsevyys TDLA

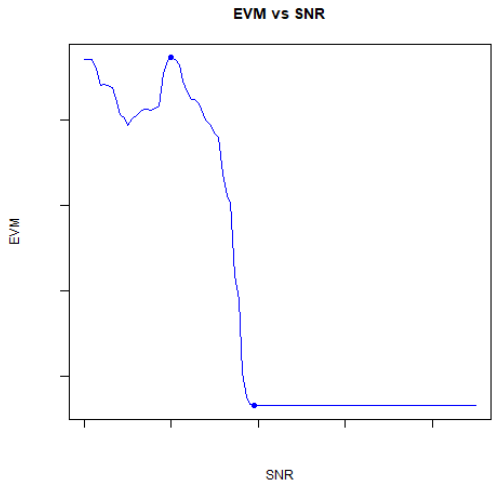


Kuva 8: Selittäjien suhteellinen merkitsevyys TDLC

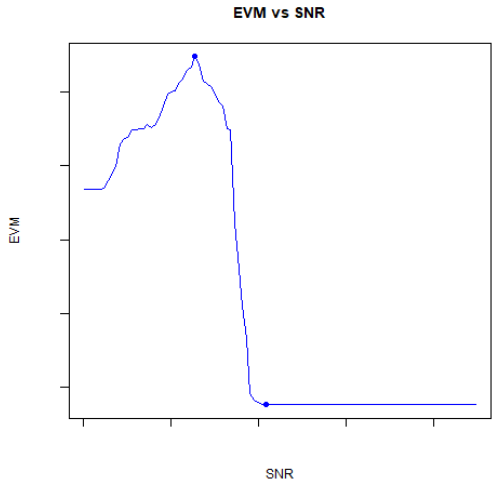
Kuten nähdään, signaalikohinasuhde on enimmäkseen selvästi muita hallitsevammassa asemassa lukuunottamatta TDLC-kanavaa, jossa nopeudella näyttäisi olevan suurin piirtein yhtä merkittävä osa tappiofunktion arvon pienentämisessä.

Seuraavat neljä kuvaa sisältävät sellaiset kahden muuttujan osittaisriippuvuuskuviot jokaiselle kanavalle, joissa toisena muuttujana on EVM (vaste),

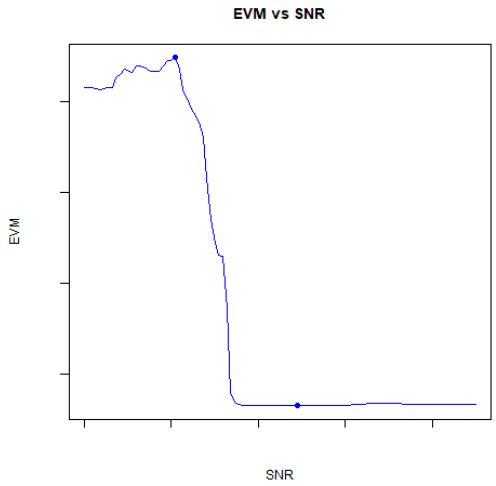
ja toisena SNR (selvästi merkityksellisin selittäjä TDLC-kanavaa lukuunottamatta). TDLC-kanavalle esitetään lisäksi sellainen osittaisriippuvuuskuvio, joka sisältää muuttujat VEL ja EVM.



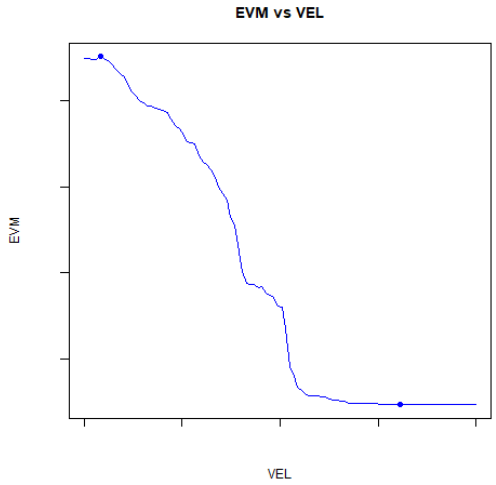
Kuva 9: AWGN



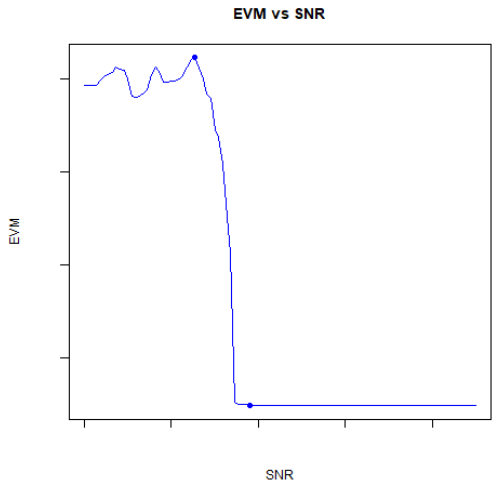
Kuva 10: EPA



Kuva 11: TDLC

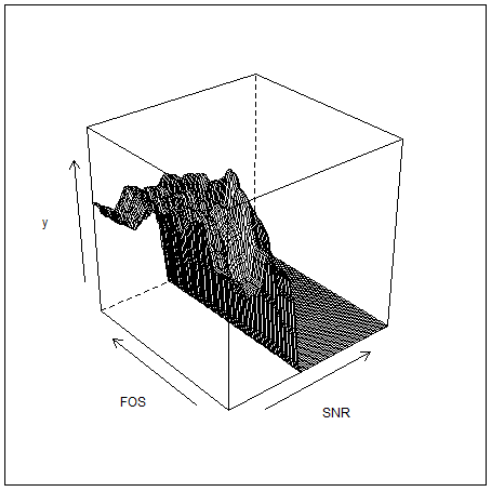


Kuva 12: TDLC

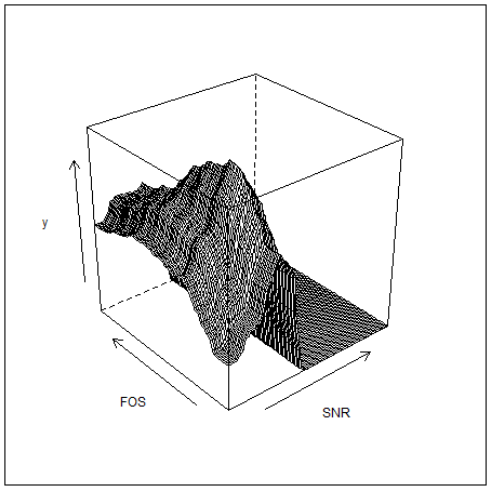


Kuva 13: TDLA

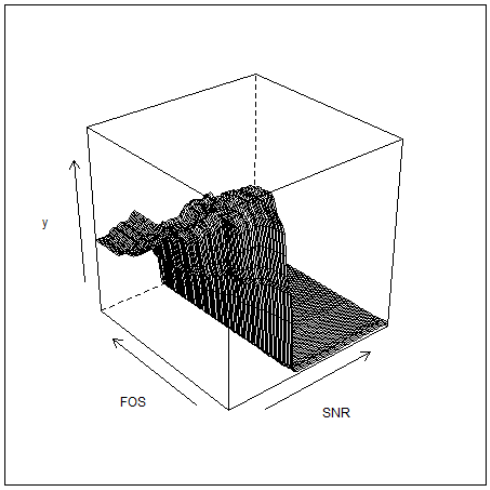
Seuraavat 12 kuvaa sisältävät kaikki sellaiset kolmen muuttujan osittaisriippuvuuskuviot jokaiselle kanavalle, joissa kaksi muuttujista on EVM ja SNR.



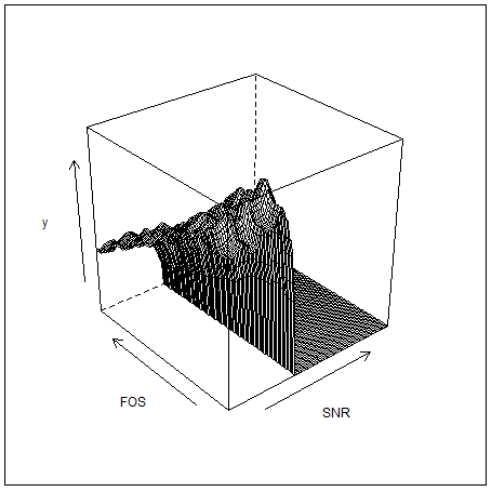
Kuva 14: AWGN



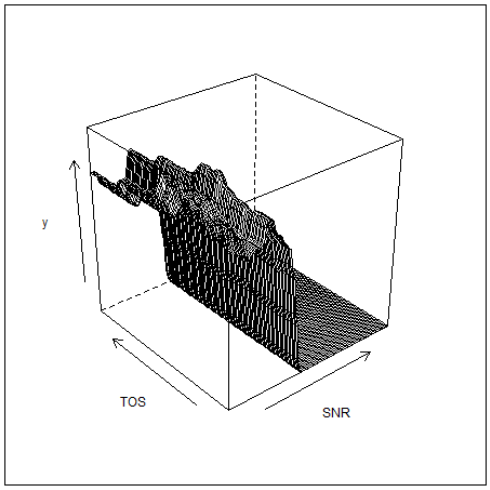
Kuva 15: EPA



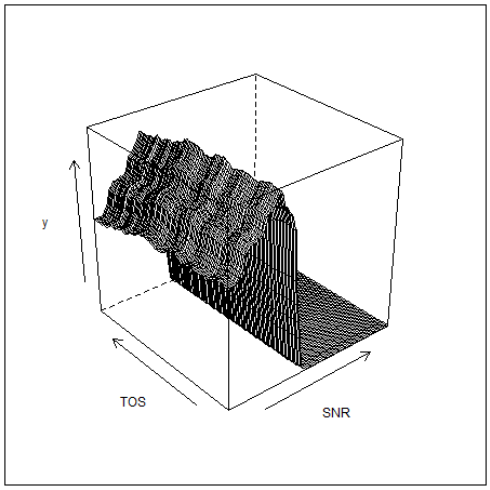
Kuva 16: TDLC



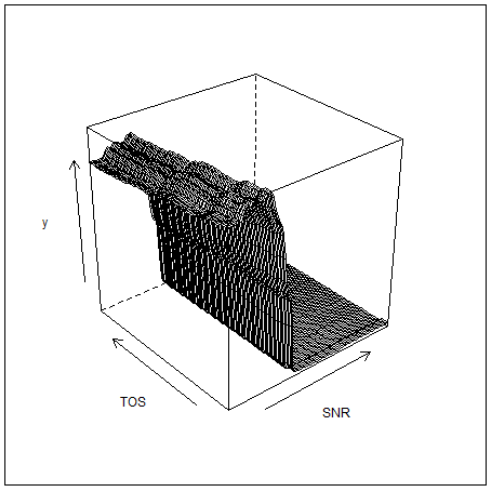
Kuva 17: TDLA



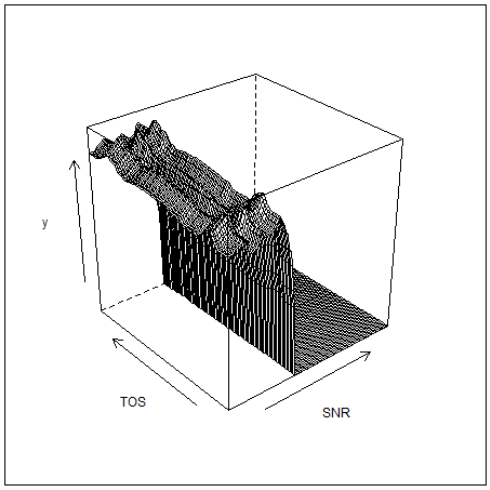
Kuva 18: AWGN



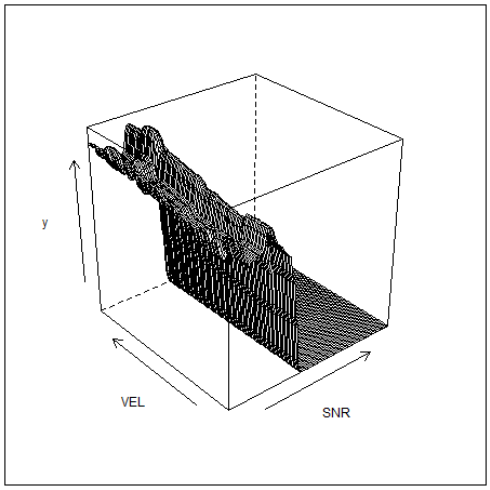
Kuva 19: EPA



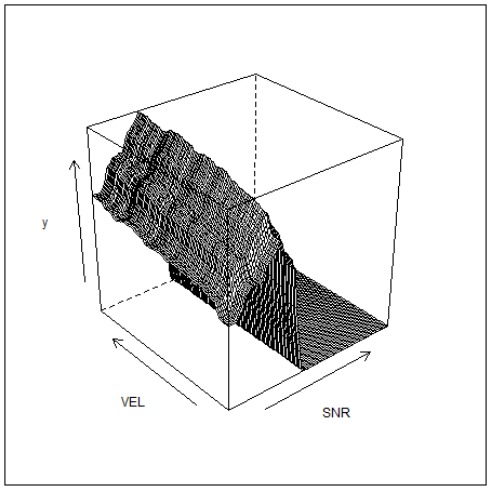
Kuva 20: TDLC



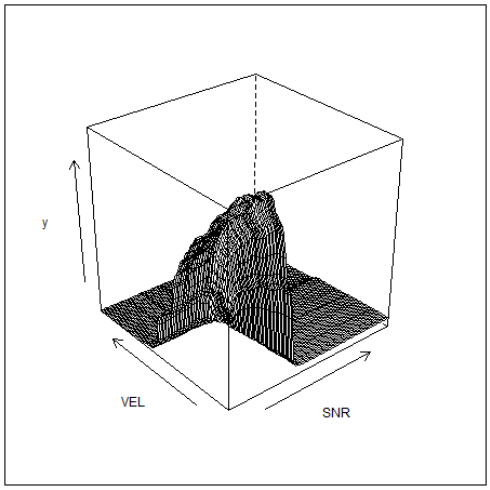
Kuva 21: TDLA



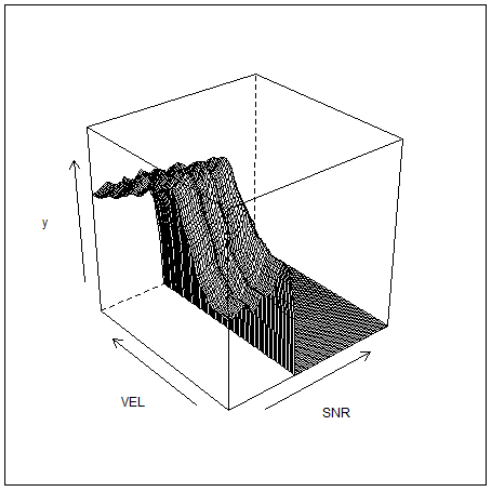
Kuva 22: AWGN



Kuva 23: EPA

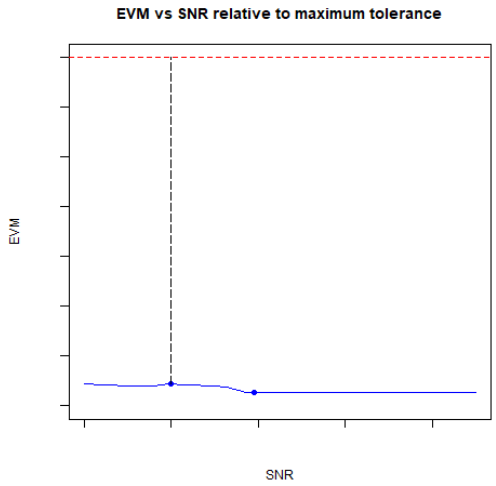


Kuva 24: TDLC

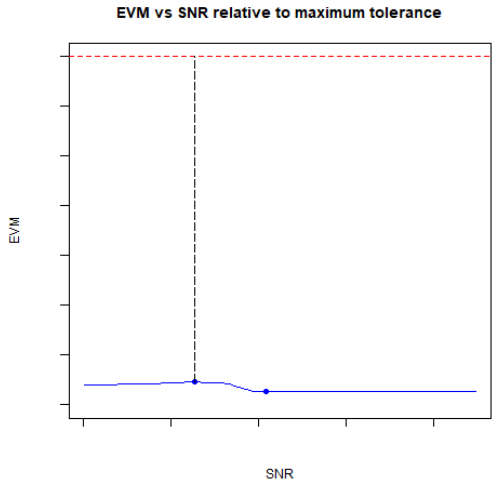


Kuva 25: TDLA

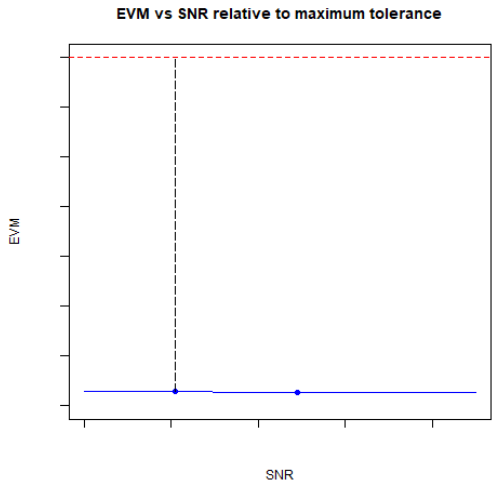
Seuraavat neljä kuvaa sisältävät samat kahden muuttujan osittaisriippuvuuskuviot kuin aiemmin, mutta eri skaalalla; nyt kuvaan on merkitty yläraja, mitä EVM ei saa ylittää.



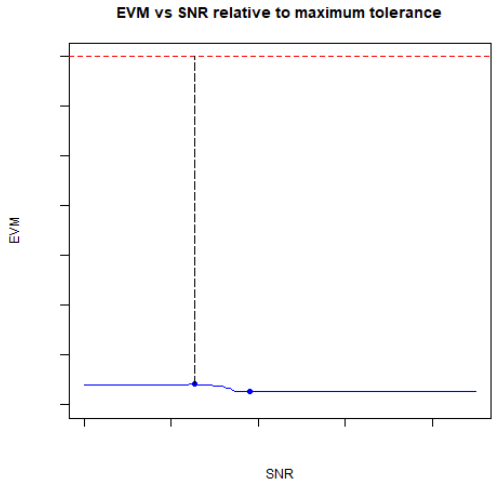
Kuva 26: AWGN



Kuva 27: EPA



Kuva 28: TDLC



Kuva 29: TDLA

Tulosten perusteella vaikuttaisi siltä, että signaali-kohinasuhde on TDLC-kanavaa lukuunottamatta ainoa parametri, jolla voisi havaita olevan vasteeseen minkäänlaista systemaattista vaikutusta. Tämän takia tuloksista on

TDLC-kanavaa lukuunottamatta jätetty kokonaan pois sellaiset osittaisriippuvuuskuviot, joissa signaali-kohinasuhde ei ole mukana. Näyttää edelleen siltä, että signaali-kohinasuhteelle on olemassa kynnysarvo, joka eri kanavilla löytyy suurinpiirtein samasta kohdasta. EVM pysyy jotakuinkin vakiona tämän kynnyksen molemmin puolin, ja tämä vakio on kynnyksen alapuolella hieman suurempi kuin kynnyksen yläpuolella.

Mielenkiintoisen poikkeuksen muodostaa TDLC-kanava, jossa nopeudella vaikuttaisi olevan likimäärin saman suuruinen merkitys vasteen selittäjänä kuin signaali-kohinasuhteella, ja vieläpä sellainen merkitys, että EVM pienenee nopeuden kasvaessa, kunnes se tiettyä arvoa suuremmilla nopeuksilla näyttäisi pysyvän suurin piirtein vakiona. Tapahtuuko näin todella, vai onko kyse simulaattorin toiminnasta, vai ehkäpä jostain sekoittavasta tekijästä, jota malli ei ota huomioon, jostain nopeuden kanssa korreloivasta muuttujasta joka kokonaan puuttuu mallista? Allekirjoittaneen arvaus on näistä viimeinen vaihtoehto, mutta tämän varmistaminen vaatisi lisätutkimuksia.

5 Lopuksi

Työssä on esitelty päätöspuiden teoriaa, vahvistamista yleisemmällä tasolla, ja päätöspuiden vahvistamista. Mielenkiintoisia ja mielestäni oleellisia kysymyksiä vahvistamiseen liittyen ovat: Miksi, miten, kuinka hyvin ja millä ehdoilla se toimii?

AdaBoostin on havaittu pienentävän tuloksena saatavan luokittelijan testivirhettä usein senkin jälkeen, kun opetusvirhe on painunut nollaan. Tämän syyksi Schapire, Freund, Bartlett ja Lee esittävät sen, että AdaBoost kasvattaa marginaalien jakaumaa [12]. Breimanin mukaan tämä ei ole ainakaan tyhjentävä selitys, ja hän pyrkii esittämään väitteensä tueksi algoritmin, jonka testivirhe ei ole niin hyvä kuin sen pitäisi marginaalien käyttäymisen perusteella olla [2], [4]. Hän myös esittää oman selityksensä, jota on käsitelty luvussa 3.8, ja jonka pohjana on [2]. Reyzin ja Schapiren [10] mukaan Breimanin vastaesimerkkinä käyttämä algoritmi on AdaBoostia monimutkaisempi, ja jos katsotaan marginaalien koko jakaumaa, eikä ainoastaan marginaalien minimiä, Breimanin kokeet eivät käy vastaesimerkistä Schapiren, Freundin, Bartlettin ja Leen teorialle. Keskustelu marginaalien merkityksestä vahvistamisessa on tästä edelleen jatkunut, mm. [8].

Viitteet

[1] Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. (1984). *Classification and Regression Trees* Monterey, CA: Wadsworth and Brooks/Cole

[2] Breiman, Leo. (1997). Arcing the Edge, *Tech. rep. 486, Statistics Department, University of California at Berkeley.*

[3] Breiman, Leo. (1998). Arcing classifiers (with discussion), *Annals of Statistics* **26**: 801-849.

[4] Breiman, Leo. (1999). Prediction games and arcing algorithms, *Neural Computation* pp. 1493-1517.

[5] Friedman, Jerome H. (1999). Stochastic gradient boosting, *technical report*, Stanford University.

[6] Friedman, J., Hastie T. and Tibshirani R. (2000). Additive logistic regression: a statistical view of boosting (with discussion), *Annals of Statistics* **28**: 337-407.

[7] Friedman, Jerome H. (2001). Greedy Function Approximation: A Gradient Boosting Machine, *Annals of Statistics* **29**(5): 1189-1232.

[8] Gao, W., Zhou Z. (2013). On the doubt about margin explanation of boosting, *Artificial Intelligence* **203** (2013): 1-18.

[9] Hastie T., Tibshirani R., Friedman J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*, *Springer Series in Statistics*, Corrected 12th printing - Jan 13, 2017

[10] Reyzin, L., Schapire R. E. (2006). How boosting the margin can also boost classifier complexity, *Proceeding of 23rd International Conference on Machine Learning*, Pittsburgh, PA, pp. 753–760.

[11] Ripley, Brian D. (1996). *Pattern Recognition and Neural Networks*, Cambridge University Press.

[12] Schapire R. E., Freund Y., Bartlett P., Lee W. S.(1998) *Boosting the margin: A new explanation for the effectiveness of voting methods The Annals of Statistics* **26**(5), 1651-1686.