Viikko 4: Päätöspuita ja mallinvalintaa

Matti Kääriäinen

matti.kaariainen@cs.helsinki.fi

Exactum C222, 19-21.11.2008.

Tällä viikolla

Sisältösuunnitelma:

- Päätöspuiden perusteet
- Testaamista ja mallinvalintaa
- Kertausta (jos aikaa jää)

Ennustusongelma

Tarkastellaan taas luokittelua:

- $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$, missä $\mathcal{X}_i = \mathbb{R}$ tai diskreetti
- $\mathcal{Y}$ luokkien joukko, $|\mathcal{Y}| \geq 2$
- Tappiofunktiona 0/1-tappio $L_{0/1}$

Koska $\mathcal{X}$ ei ole jatkuva eikä diskreetti, ennustusongelmaa ei voi aiemmin esitetyistä luokittelumenetelmistä ratkaista kuin lähin naapuri -menetelmillä. Tällöinkin hyvä diskreetit ja jatkuva-arvoiset syötekomponentit yhdistävä etäisyysfunktio voi olla hankala löytää.

Mikä neuvoksi? Päätöspuut!

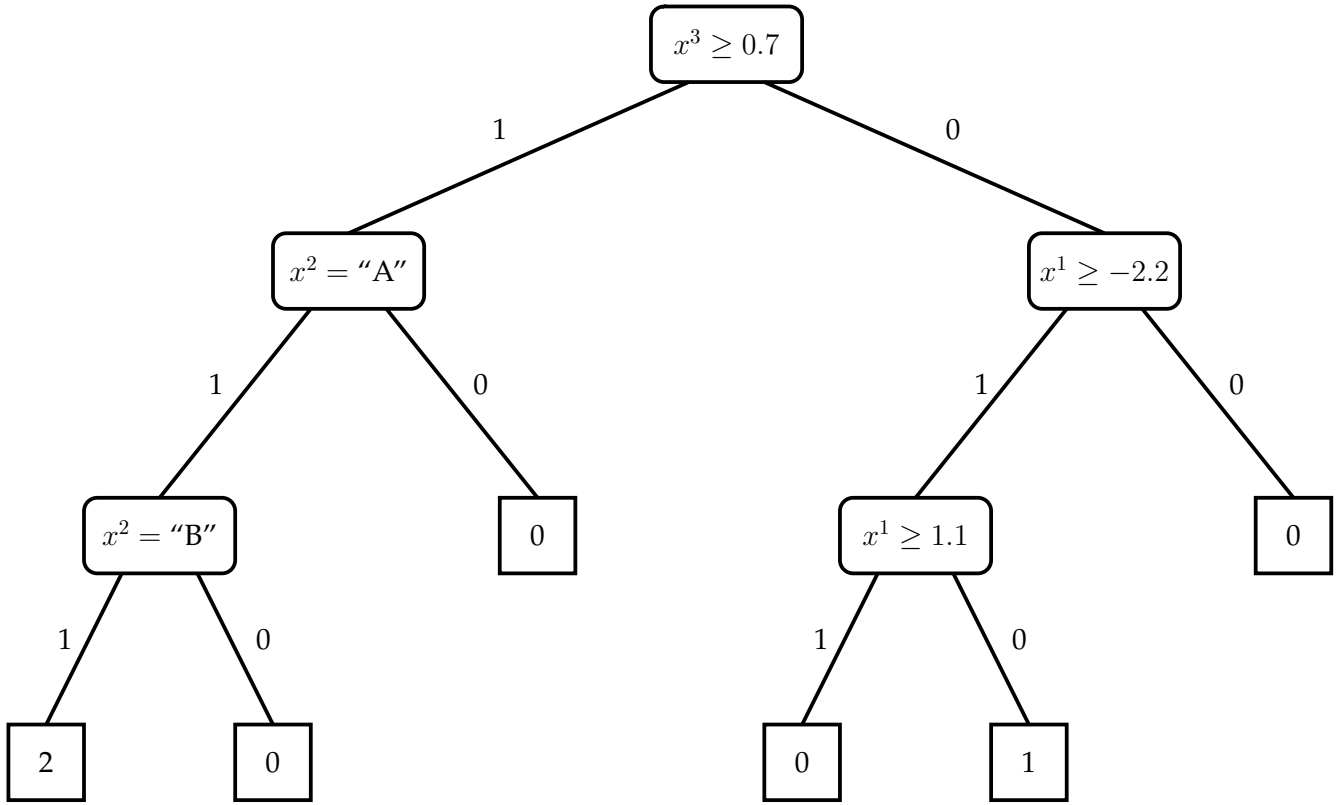
Päätöspuut

Päätöspuun rakenne:

- Esitetään luokittelija binäärisenä puuna T
- Puun T kuhunkin sisäsolmuun v liittyy testi $\text{Test}_v : \mathcal{X} \to \{0, 1\}$. Solmun v lapset v_0 ja v_1 on nimetty testin tuloksin.
- Puun jokaiseen lehteen liittyy ennuste (tässä luokka $y \in \mathcal{Y}$)

Testien voi ajatella olevan (yksinkertaisia) ennustajia, jotka ennustavat, kumpaan alipuuhun syöte tulee jatkokäsittelyä varten ohjata.

Päätöspuu kuvana



Testeistä

Rajoitutaan seuraavassa kahdenlaisiin yksinkertaisiin testeihin, joissa molemmissa testataan yhtä syötekomponenttia kerrallaan:

- $\mathcal{X}_i = \mathbb{R}$: Testataan, päteekö $x^i \geq \theta$, missä $\theta \in \mathbb{R}$ on testin kynnysparametri. Jos $x^i \geq \theta$ testi saa arvon 1, muuten arvon 0.
- $\mathcal{X}_i$ diskreetti: Testataan, päteekö $x^i = \tilde{x}^i$, missä $\tilde{x}^i \in \mathcal{X}^i$ on testissä käytettävä vertailukohde. Jos $x^i = \tilde{x}^i$ testi saa arvon 1, muuten arvon 0.

Sisäsolmun testi määräytyy siis kahdesta tekijästä: mitä syötteen komponenttia $i = 1, \ldots, d$ testataan, ja mihin sitä verrataan (θ tai $\tilde{x}^i$).

Päätöspuu luokittelijana

Päätöspuu T luokittelee syötteen $x \in \mathcal{X}$ kutsumalla rekursiivista funktiota `Luokittele`(juuri(T), x), joka määritellään seuraavasti:

`Luokittele`(solmu v, syöte x)
  jos v on lehtisolmu
    palauta v:hen talletettu luokka y
  muuten
    palauta `Luokittele`($v_{\mathsf{Test}_v(x)}$, x)

Funktio `Luokittele` ohjaa siis syötteen x puun T juuresta testien määräämää polkua pitkin johonkin T:n lehteen, johon talletettu luokka määrää T:n x:lle antaman luokituksen. Näin määräytyvä funktio on T:n määrittelemä luokittelija $f_T : \mathcal{X} \to \mathcal{Y}$.

Päätöspuiden oppiminen

- Optimaalisen* päätöspuun oppiminen on laskennallisesti vaikea ongelma (NP-täydellinen, vaikea approksimoida).
- Käytännössä päätöspuita voi kuitenkin oppia melko yksinkertaisella kaksivaiheisella strategialla:
  - Kasvatetaan opetusdatan mahdollisimman hyvin luokitteleva puu ahneella heuristiikalla
  - Karsitaan puusta osia pois ylisovittamisen estämiseksi
- Syy kaksivaiheeseen strategiaan on empiirinen: käytäntö on osoittanut, että kannattaa ensin kasvattaa opetusdatalla tarkka iso ja monimutkainen puu ja sitten karsia sitä, eikä yrittää kasvattaa sopivan kokoista puuta suoraan.

*Esim. solmujen lukumäärältään pienimmän opetusdatan oikein luokittelevan

Päätöspuun kasvattaminen

Päätöspuu kasvatetaan rekursiivisesti seuraavalla ahneella kasvatusheuristiikalla, alkaen kutsusta `Kasvata`(S), missä S on kaiken kasvatusdatan joukko.

`Kasvata`(opetusdatan (osa)joukko S)
  jos `Lopetusehto`(S)
    luo lehti, johon liitetty S:n yleisin luokka
  muuten
    luo uusi sisäsolmu v, jolle Test$_v$ = `ValitseTesti`(S)
    v_0 = `Kasvata`($\{x \in S \mid \texttt{Test}_v(x) = 0\}$)
    v_1 = `Kasvata`($\{x \in S \mid \texttt{Test}_v(x) = 1\}$)

`Kasvata` jakaa siis opetusdataa testien perusteella rekursiivisesti osajoukkoihin, kunnes lopetusehto täyttyy.

Päätöspuun kasvattamisalgoritmin idea

- Kasvatusalgoritmi valitsee funktion `ValitseTesti`(S) avulla testit, jotka jakavat datan pienempiin ja pienempiin osajoukkoihin juuresta lehtiä kohti kuljettaessa.
- `Lopetusehto`(S) testaa, voidaanko rekursio jo lopettaa. Tyypillisesti testataan, onko S riittävän luokkahomogeeninen tai pieni, esim. onko S kokonaan samaa luokkaa tai alle 5 alkioinen.
- Puuttuva palanen: Miten `ValitseTesti`(S) valitsee testit? Tavoitteena tietysti, että valitut testit yhdessä jakaisivat syötteet mahdollisimman vähin testein luokkahomogeenisiin osajoukkoihin, mutta miten tähän päästään?

Entropia

- Eräs yleinen homogeenisuuden (tai sen puutteen) mitta on entropia, joka opetusesimerkkien joukon S luokkajakaumaan sovellettuna saa muodon

$$H(S) = -\sum_{y' \in \mathcal{Y}} \frac{|\{(x,y) \in S \mid y = y'\}|}{|S|} \log \left(\frac{|\{(x,y) \in S \mid y = y'\}|}{|S|} \right).$$

- $H(S) = 0$, jos S:ssä on vain yhtä luokkaa.
- $H(S) = \log |\mathcal{Y}|$, jos S:n luokkajakauma on maksimaalisen epähomogeeninen eli S:ssä on kaikkia luokkia yhtä paljon.
- Muuten $H(S)$ on jotain tältä väliltä, ollen sitä pienempi mitä luokkahomogeenisempi S on.

Informaatiolisä (Information Gain)

- Mitataan testin hyvyyttä informaatiolisällä

$$\mathsf{IG}(\mathsf{Test}, S) = H(S) - \left(\frac{|S_0|}{|S|} H(S_0) + \frac{|S_1|}{|S|} H(S_1) \right),$$

  missä $S_i = \{x \in S \mid \mathsf{Test}(x) = i\}$, $i = 0, 1$.

- Informaatiolisän kaavassa termi $H(S)$ mittaa, kuinka paljon luokkamuuttujassa on entropiaa ennen testiä Test, ja termi $(|S_0|/|S|H(S_0) + |S_1|/|S|H(S_1))$ mittaa, kuinka paljon entropiaa on (odotusarvoisesti) jäljellä testin jälkeen.

- Heuristiikka ValitseTesti(S): Valitaan testi, jolle $\mathsf{IG}(\mathsf{Test}, S)$ on suurin, eli joka sovellettuna opetusdatan joukkoon S antaa suurimman informaatiolisän.

Päätöspuun kasvatusalgoritmin käytännön toteutuksesta

- Algoritmin `Kasvata`(S) ainoa laskennallisesti epätriviaali osa on hyvän testin valinta.
- Tämäkään ei ole kovin vaikeaa, sillä erilaisen tuloksen tuottavia testejä on vain (pienehkö) äärellinen määrä:
  - Jos $\mathcal{X}_i$ on diskreetti, erilaisia vertailukohteita $\tilde{x}^i$ on $|\mathcal{X}_i|$ kappaletta
  - Jos $\mathcal{X}_i$ on jatkuva, eri tavalla opetusdatan jakavia kynnysarvoja θ on korkeintaan $|S| + 1$ kappaletta

  Parhaan testin voi siis valita laskemalla informaatiolisän kaikkien syötekomponenttien kaikille (erilaisen tuloksen tuottaville) testeille, ja valitsemalla näistä parhaan.

Väliaikaraportti

- Nyt on selvillä, miten opetusdataan hyvin sopiva päätöspuu voidaan kasvattaa.
- Seuraavaksi vuorossa karsintavaihe, jonka tehtävänä on
  - parantaa puun odotusarvoista tappiota
  - pienentää puuta (tehokkuus ja ymmärrettävyyssyistä)
- Karsintavaiheessa pyritään siis karsimaan kasvatetusta puusta pois sellaiset osat, jotka pienentävät opetusvirhettä mutta eivät ole hyödyllisiä odotusarvoisen tappion kannalta.

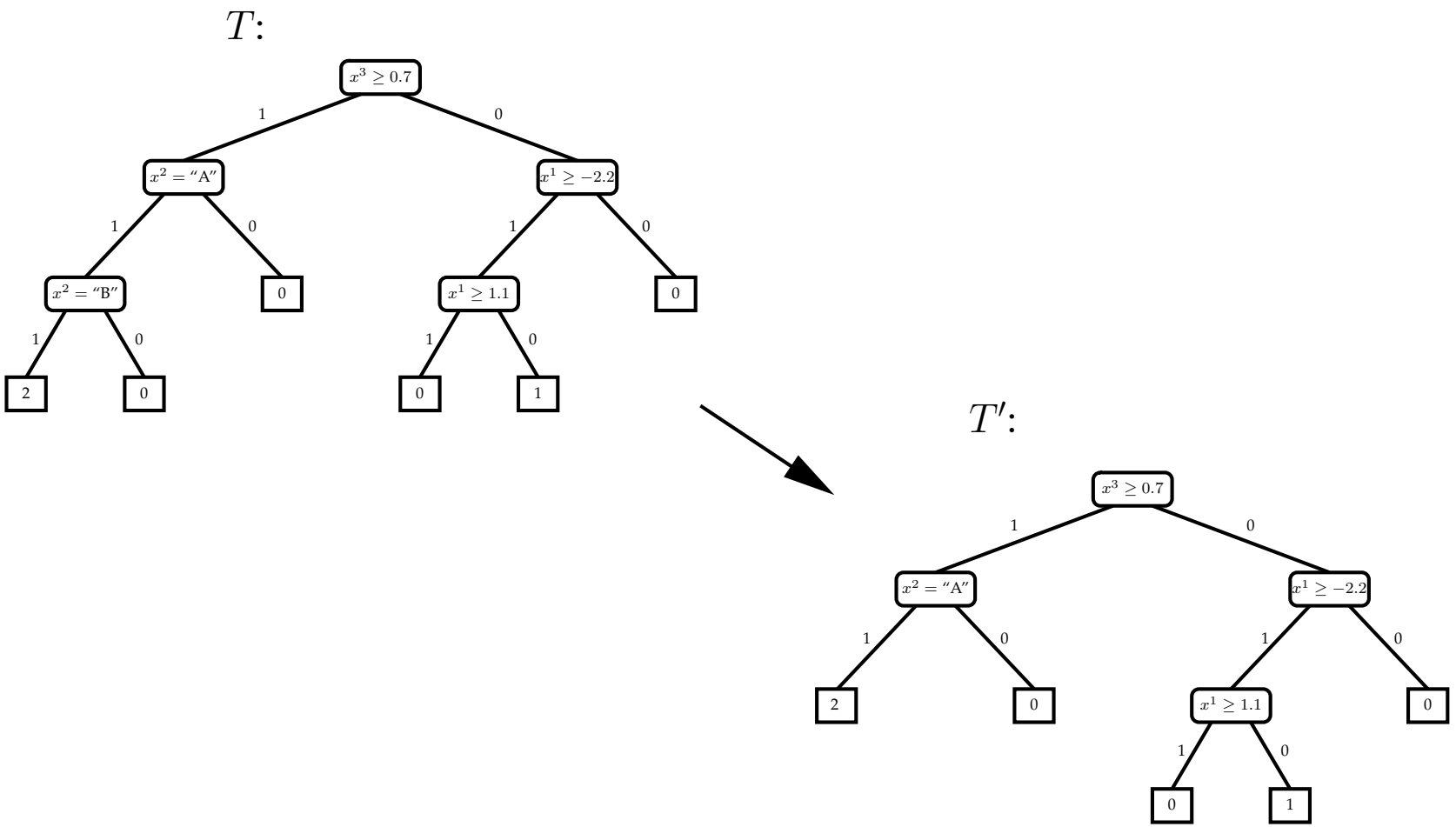
Päätöspuun karsinta

Päätöspuun T karsinta T' on päätöspuu, joka täyttää seuraavat ehdot:

1. T' on T:n alipuu, jolla on sama juuri kuin T:llä
2. T':n sisäsolmujen testit ovat samat kuin T:n vastaavissa sisäsolmuissa

Proseduraalisesti: T' saadaan T:stä korvaamalla joitain T:n alipuita uusilla lehdillä.

Karsinta kuvana



Karsintastrategia (Reduced Error Pruning)

- Jaetaan alkuperäinen opetusdata S satunnaisesti kahteen osaan S_{grow} ja S_{prune}. Valitaan siis esim. 70% opetusdatasta joukkoon S_{grow}, ja jätetään loput 30% joukkoon S_{prune}.
- Kasvatetaan päätöspuu T käyttäen opetusdatan joukkona kasvatusdatan joukkoa S_{grow}
- Karsintavaiheessa etsitään T:n karsinta T', jolle
  1. Minkään T:n karsinnan virhe karsintadatan joukon S_{prune} esimerkeillä ei ole pienempi kuin T':n virhe ko. esimerkeillä
  2. T' on solmujen lukumäärältään pienin ehdon 1. toteuttavista T:n karsinnoista

Karsinta-strategian idea

- Ylisovittamisen estämiseksi vain osa opetusdatasta (S_{grow}) käytetään päätöspuun T kasvattamiseen
- Loppu data (S_{prune}) käytetään T:n “parhaan” (eli karsintadatalla pienivirheisimmän ja pienimmän) karsinnan löytämiseen
- Koska T:n karsintojen joukko on paljon pienempi ja yksinkertaisempi kuin kaikkien mahdollisten päätöspuiden joukko, ylisovittamisen vaara pienenee.
- Strategian haittapuolena on se, että kaikkea opetusdataa ei voida käyttää puun kasvattamiseen.

Karsintaa algoritmina

- Edellä hahmoteltu karsintastrategia voidaan toteuttaa tehokkaasti käymällä karsittava puu T läpi “kokoavasti” (bottom-up)
- Läpikäytävät T:n alipuut $\tilde{T}$ ovat kahta eri tyyppiä:
  - **Tapaus 1:** $\tilde{T}$ on pelkkä lehti. Tällöin $\tilde{T}$:n paras karsinta saadaan valitsemalla lehteen karsintadatalla pienimmän virheen tuottava luokka.
  - **Tapaus 2:** $\tilde{T}$ ei ole pelkkä lehti. Tällöin $\tilde{T}$ koostuu juurena toimivasta sisäsolmusta ja sen alipuista $\tilde{T}_0$ ja $\tilde{T}_1$. Tällöin $\tilde{T}$:n paras karsinta on pelkkä lehti tai karsinta, joka saadaan korvaamalla alipuut $\tilde{T}_0$ ja $\tilde{T}_1$ parhailla karsinnoillaan $\tilde{T}'_0$ ja $\tilde{T}'_1$.
- Täsmällinen formulointi, todistus, ja toteutus: Harjoitustehtävä. . .

Päätöspuut: yhteenveto

Päätöspuuoppimista pidetään usein vanhahtavana adhoc-lähestymistapana, mutta sillä on monia hyviä puolia:

- Yleiskäyttöisyys – diskreetit ja jatkuva-arvoiset piirteet ja moniluokkaisuus eivät tuota ongelmia
- Ymmärrettävyys – ihmisasiantuntijat kokevat pystyvänsä ymmärtämään päätöspuuta tarkastelemalla syitä luokittelupäätösten takana
- Tehokkuus – päätöspuiden oppimisalgoritmit ja päätöspuut luokittelijoina suhteellisen nopeita
- Tarkkuus – vaikka päätöspuut eivät yleensä ole kaikkein tarkimpia ennustajia, ne ovat usein melko tarkkoja

Testaus ja mallinvalinta

Kertausta:

- Toistaiseksi esitetty
  - Oppimisen malli ja tavoite
  - Lukuisia koneoppimismenetelmiä tavoitteen saavuttamiseen
- Kysymyksiä:
  - Mistä tietää, kuinka hyvin tietty koneoppimismenetelmä ratkaisee tietyn ennustusongelman?
  - Kuinka valita “hyvä” tai “paras” koneoppimismenetelmä tiettyyn ennustusongelmaan?

Seuraavassa esitettävät vastaukset: Testaus ja mallinvalinta

Tehtävä

Tarkastellaan seuraavaa tehtävää:

- Annettuna: Ennustusongelma, opetusdataa ongelmasta, ja koneoppimismenetelmä (parametreineen, jos sellaisia on)
- Tehtävä: Arvioi, kuinka hyvin koneoppimismenetelmä toimii annetussa ongelmassa, ts. mikä on sen tuottaman ennustajan odotusarvoinen tappio?

Kuten muistetaan, kysymyksen tekee epätriviaaliksi se, ettei jakaumaa D jonka suhteen odotusarvoista tappiota mitataan tunneta.

Testaaminen opetusdatalla

- Yksi tapa arvioida opitun ennustajan f odotusarvoista tappiota on käyttää sen arviointiin opetusvirhettä eli tappiota opetusdatalla:

$$\mathrm{Err}_{\mathrm{train}}(f) = \frac{1}{|S|} \sum_{(x,y) \in S} L(y, f(x))$$

- Tämä strategia on kuitenkin yleensä **huono**, ja johtaa helposti ylioptimistiseen arvioon – f on opittu samasta datasta jolla sitä testataan, joten mahdollista ylisovittamista ei havaita!

Tarvitaan siis parempi strategia.

Testaaminen erillisellä datalla

- Jaetaan opetusdata satunnaisesti kahteen osaan: Varsinaiseen opetusdataan S_{train}, ja testidataan S_{test}.
- Tuotetaan koneoppimismenetelmällä ennustaja f opetusdatasta S_{train}
- Lasketaan ennustajan f testivirhe eli tappio testidatalla

$$\text{Err}_{\text{test}}(f) = \frac{1}{|S_{\text{test}}|} \sum_{(x,y) \in S_{\text{test}}} L(y, f(x))$$

- Käytetään testivirhettä $\text{Err}_{\text{test}}(f)$ arviona f:n odotusarvoisesta virheestä $\mathbb{E}_D[L(y, f(x))]$.

Miksi käytetään erillistä testidataa?

- Koska f on opittu vain joukosta S_{train}, se on riippumaton testiesimerkeistä S_{test}.
- Näin ollen f:n testivirhe $\text{Err}_{\text{test}}(f)$ on jakautunut samoin kuin f:n "todellinen" virhe samankokoisella otoksella jakaumasta D valittavia toistaiseksi näkemättömiä syötteitä.
- Testivirhettä on siis harhaton estimaatti f:n odotusarvoisesta virheestä.
- Testivirhe voidaan kuitenkin laskea suoraviivaisesti, koska testiesimerkkeihin liittyvät "oikeat vastaukset" tunnetaan.

Erillisen testidatan hyviä ja huonoja puolia

Trade-off:

- Testivirhe on sitä tarkempi estimaattori odotusarvoiselle virheelle mitä enemmän testidataa on käytettävissä.
- Koneoppimismenetelmät tuottavat tyypillisesti sitä parempia ennustajia mitä enemmän opetusdataa on käytettävissä.

Jos siis käytetään paljon dataa testaamiseen, saadaan tarkka kuva siitä kuinka hyvin oppiminen onnistuu, mutta oppiminen ei onnistu yhtä hyvin kuin jos datasta pienempi osa olisi varattu testikäyttöön. Tämä ongelma on erityisen hankala jos S on pieni.

Tyypillinen jakosuhde: esim: 90% opetukseen ja 10% testaukseen (tai 70% ja 30%).

k-kertainen ristiinvalidointi

- Jaetaan opetusdata S satunnaisesti k:hon (esim. $k = 10$) suurinpiirtein yhtä suureen osaan $S_1, \ldots, S_k$
- Opitaan k ennustajaa $f_1, \ldots, f_k$, missä f_i opitaan soveltamalla koneoppimismenetelmää opetusdatan joukkoon $S \setminus S_i$.
- Testataan ennustajaa f_i käyttäen testidatana joukkoa S_i:

$$\mathsf{Err}_{\mathsf{test}_i}(f_i) = \frac{1}{|S_i|} \sum_{(x,y) \in S_i} L(y, f_i(x))$$

- Opitaan lopullinen ennustaja f kaikesta opetusdatasta S
- Arvioidaan f:n odotusarvoista virhettä ennustajien f_i testivirheiden keskiarvolla $\frac{1}{k} \sum_{i=1}^{k} \mathsf{Err}_{\mathsf{test}_i}(f_i)$.

Ristiinvalidoinnista

Hyviä puolia:

- Ristiinvalidoinnissa lopullinen ennustaja f voidaan oppia kaikesta opetusdatasta
- Saatu estimaatti f:n todelliselle virheelle on yleensä käytännössä melko tarkka, ja sen varianssi on pienempi kuin käytettäessä $\frac{1}{k}|S|$-kokoista erillistä testidataa.

Huonoja puolia:

- Ristiinvalidointi saattaa olla laskennallisesti raskasta, sillä koneoppimisalgoritmi joudutaan ajamaan $k+1$ kertaa
- Ristiinvalidointiin pohjautuva estimaatti ei ole harhaton, eikä siitä muutenkaan osata sanoa teoreettisesti juuri mitään.

Mallinvalinta

- Koneoppimismenetelmän tehtävänä on valita ennustusongelmassa hyvin pärjäävä ennustaja
- Eri koneoppimismenetelmät (tai sama menetelmä eri parametrein) tuottavat saman ennustusongelman ratkaisuiksi erilaisia ennustajia
- Miten valita näistä paras?

Esimerkkejä mallinvalintaongelmista

Tehtävänä ratkaista kaksiluokkainen ennustusongelma, jossa piirteet ovat diskreettejä.

- malli = koneoppimismenetelmä: taulukointi, k lähintä naapuria, Naive Bayes, päätöspuu, ...
- malli = koneoppimismenetelmän parametrit: k lähintä naapuria -menetelmän etäisyysfunktio ja k, Naive Bayes -menetelmän luokkapriori, päätöspuun kasvatus- ja karsintaheuristiikka, ...
- malli = syötteen piirteet: Mitkä piirteet otetaan mukaan, miten jatkuva-arvoiset piirteet diskretoidaan, ...

Mallinvalintaongelma

Mallinvalintamenetelmän tehtävänä on siis seuraava:

- Annettuna opetusdataa ja pienehkö joukko erilaisen ennustajan tuottavia koneoppimismenetelmiä/koneoppimismenetelmän parametrisointeja
- Valittava, mitä näistä käytetään lopullisen ennustajan tuottamiseen
- Tavoite sama kuin koneoppimisessa – halutaan löytää ennustaja f, jonka odotusarvoinen tappio on pieni, mutta nyt pienemmästä koneoppimismenetelmien tuottamien kandidaattien joukosta

(Jäljelle jää vielä korkeamman tason metaongelma: Miten valita “annettu” joukko koneoppimismenetelmiä, joiden joukosta parasta lähdetään etsimään. . .)

Mallinvalintaongelman ratkaisemisesta

Idea:

- Edellä on esitetty, miten yhden koneoppimismenetelmän hyvyyttä voi testata erillisen testidatan tai ristiinvalidoinnin avulla
- Sovelletaan näitä testausmenetelmiä yksitellen jokaiseen valittavana olevaan malliin (eli koneoppimismenetelmään). Tuloksena saadaan arvio kunkin mallin hyvyydestä.
- Valitaan malli, johon liittyvä arvio on paras.

Mallinvalinta ja ylisovittaminen

- Jos valittavana olevia malleja on paljon, mallinvalintamenetelmät kärsivät ylisovittamisongelmista samaan tapaan kuin koneoppimismenetelmätkin.
- Ylisovittamisongelmien (ja laskentavaivan) minimoimiseksi valittavana olevien mallien määrä pyritään yleensä pitämään suhteellisen pienenä (muutamista satoihin tai korkeintaan tuhansiin malleihin).
- Mallinvalinnankin tulosten hyvyys on syytä testata, esim. käyttämällä mallinvalinnasta erillistä testidataa.

Toistuvasta testaamisesta

Tyypillinen tilanne käytännön koneoppimisessa ja koneoppimistutkimuksessa:

1. Jaetaan opetusdata opetus- ja testausosiin S_{train} ja S_{test}
2. Valitaan koneoppimismenetelmä ja sille hyvät parametrit ristiinvalidoinnilla opetusdatalla S_{train}
3. Opitaan “lopullinen” ennustaja f opetusdatasta S_{train}
4. Testataan ennustajaa f testidatalla S_{test}
5. **Jos f:n testivirhe ei olekaan niin pieni kuin olisi toivottu, palataan takaisin kohtaan 1. tai 2.**

Kohdan 5. johdosta vaarana ylisovittaminen – testidata ei iteroidessa enää olekaan riippumatonta lopullisesta hypoteesista f!

Yhteenveto ohjatusta oppimista

- **Tavoite:** Tuottaa automaattisesti opetusdatan perusteella ennustaja, joka antaa hyviä ennusteita opetusvaiheessa tuntemattomilla syötteillä
- **Perusoletus (iid):** Opetusdata on otos samasta jakaumasta kuin tulevat syötteet ja niihin liittyvät “oikeat vastaukset”
- Yleispätevää kaikki ongelmat automaattisesti ratkaisevaa koneoppimismenetelmää ei ole olemassa, mutta moniin ennustusongelmiin voidaan tuottaa hyviä ratkaisuja jo kurssilla tutuiksi tulleilla koneoppimismenetelmillä
- Hyvän koneoppimismenetelmän valinta (tai suunnittelu) tiettyyn ongelmaan on monimutkainen taiteen laji, jossa testaus- ja mallinvalintatekniikoista on apua.