Hahmontunnistuksen perusteet

T-61.231 (3 ov) L

Syksy 2001

Luennot: **Vuokko Vuori**
Laskuharjoitukset: **Matti Aksela**

1. FOREIGN STUDENTS

Lectures and slides are in Finnish

Exercises are held in Finnish but the problems and their solutions are published in English. Course assistant does speak English

To pass this course you need to pass the exam and do the course assignment

For more information, see
http://www.cis.hut.fi/Opinnot/T-61.231/index2001_en.html

2. YLEISTÄ KURSSISTA

2.1 Kurssin suorittaminen

Kurssin suorittaminen sisältää pakollisen harjoitustehtävän ja tentin.

2.2 Ilmoittautuminen

Ilmoittautukaa kurssille www-topin avulla.

2.3 Tiedotukset

Kurssista tiedotetaan webissä http://www.cis.hut.fi/Opinnot/T-61.231, ryhmässä news://nntp.tky.hut.fi/opinnot.tik.informaatiotekniikka sekä Informaatiotekniikan laboratorion ilmoitustaululla kolmannen kerroksen aulassa B-käytävän suulla.

2.4 Luennot

Luennot pidetään torstaisin kello 10–12 salissa T2.

Luennoitsija erikoisopettaja DI Vuokko Vuori (mailto:vuokko.vuori@hut.fi).

Luentokalvot ovat luennon jälkeen nähtävillä osoitteessa http://www.cis.hut.fi/Opinnot/T-61.231/kalvot.pdf.

Luennoitsijan vastaanotto ennen luentoa torstaisin kello 9–10 huoneessa C310. Muista ajoista mahdollista sopia sähköpostitse.

2.5 Laskuharjoitukset

Laskuharjoitukset maanantaisin kello 14–16 salissa T3 alkaen 24.9.2001.

Harjoitukset pitää DI Matti Aksela (mailto:matti.aksela@hut.fi).

Harjoitustehtävät ovat jo ennakkoon nähtävillä osoitteessa http://www.cis.hut.fi/Opinnot/T-61.231/laskarit2001.html.

2.6 Kirja

Kurssikirjaksi on kaksi vaihtoehtoa:

1) Sergios Theodoridis, Konstantinos Koutroumbas:
Pattern Recognition, Academic Press, 1998.

2) Robert Schalkoff:
Pattern Recognition – Statistical, Structural and Neural Approaches,
John Wiley & Sons, 1992.

1. kirjan hinta eri kaupoissa:
http://www.bol.fi/fi/ 480 markkaa
http://www.harcourt-international.com/ noin 40 brittipuntaa (sis. kuljetus)
http://www.amazon.com/ noin 60 US dollaria
http://www.amazon.co.uk/ noin noin 40 brittipuntaa

2. kirjan hinta eri kaupoissa:
http://www.amazon.com/ noin 149 US dollaria
http://www.amazon.co.uk/ noin 44 brittipuntaa
Yliopiston kirjakauppa, TKK 440 markkaa

Tutustumiskappaleet ovat nähtävillä Informaatiotekniikan laboratorion kirjastossa ja laboratorion sihteerin Tarja Pihamaan huoneessa B326 olevassa harmaassa peltisessä vetolaatikostossa.

2.7 Luentomonisteet

Laskuharjoitukset ratkaisuineen ja luentokalvot ilmestyvät opetusmonisteina.

Materiaalia voi myös lainata luennoitsijalta ja assarilta itse kopioitavaksi.

Vapaaehtoinen kurssitoimittaja?

2.8 Suhde vanhaan Tik-61.131-kurssiin

Kurssi korvaa vanhan samannimisen kurssin Tik-61.131, jonka laajuus oli 2,5 ov ja joka ei sisältänyt pakollista harjoitustyötä,

2.9 Tentti

Tenttejä järjestetään kaksi: ensimmäinen 11. joulukuuta ja toinen 4. helmikuuta. Lisäksi vuoden 2002 syksyn alussa järjestään tentti.

Tentissä on 4 tehtävää à 6 pistettä, maksimi 24 pistettä. Yksi tehtävistä on essee, yksi koostuu useista pienistä sanallisista tehtävistä ja loput ovat laskutehtäviä.

Tentissä saa olla mukana matemaattinen kaavakokoelma ja tavallinen funktiolaskin.

2.10 Harjoitustehtävä

Kurssin suoritukseen kuuluu pakollinen harjoitustehtävä, joka arvostellaan hyväksytty/hylätty-periaatteella. Jos haluaa kurssista suoritusmerkinnän joulukuun tenttitulosten yhteydessä, harjoitustehtävä on saatava hyväksytysti läpi joulukuun 1. päivään mennessä.

Myöhempiinkään tentteihin ei saa osallistua, ellei harjoitustehtävä ole hyväksytysti suoritettu.

Harjoitustehtävän aiheet ovat esillä osoitteessa
http://www.cis.hut.fi/Opinnot/T-61.231/harjtyo2001.html.

Harjoitustyöhön liittyvissä asioissa ottakaa yhteyttä kurssin assistenttiin (mailto:matti.aksela@hut.fi).

3. JOHDANTO

3.1 Mitä on hahmontunnistus?

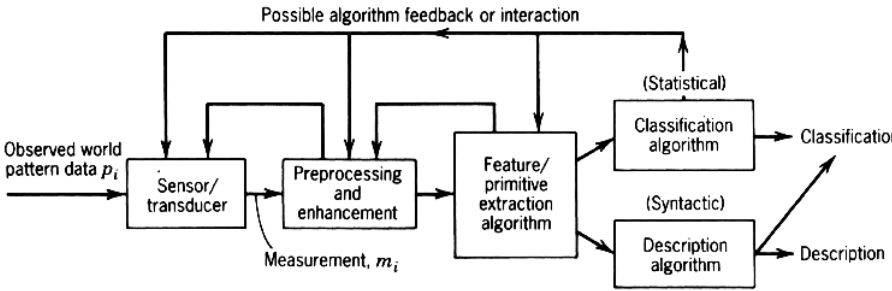
- Hahmontunnistus on tieteenala, jossa pyritään luokittelemaan tehtyjä havaintoja (hahmot) erilaisiin kategorioihin (luokat)
- Keskeisiä ongelmia ovat tiedon (mittaukset, havainnot) pelkistys (esikäsittely), tiedon kuvaus (piirreirrotus), ja tiedon luokittelu tai muu kuvaus (tunnistus)
- Hahmontunnistukseen liittyviä tieteenaloja: digitaalinen signaalin käsittely, tekoäly, neuraalilaskenta, optimointioppi, estimointiteoria, sumea logiikka, strukturaalinen mallintaminen, formaalit kielet, kieliteknologia

3.2 Sovelluksia

- Digitaalisten kuvien käsittely, tietokonenäkö, esim. satelliittikuvien tulkinta
- Puheen tunnistus, painetun tekstin ja käsialan tunnistus
- Henkilöiden tunnistus puheen, sormenjälkien, nimikirjoituksen tai silmän iriksen perusteella
- Lääketieteellisten mittausten analysointi
- Tietokoneavustettu päätöksenteko
- Laadunvalvonta, lajittelu, esim. reikä paperirainassa, palautuspullojen käsittely.
- Postin ja pankkitositteiden automaattinen käsittely
- ... ja vaikka mitä muuta!

3.3 Tunnistusjärjestelmä

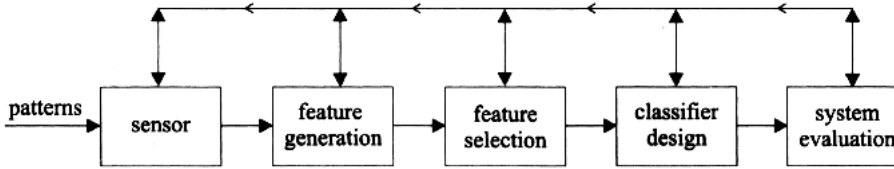
Tyypillinen tunnistusjärjestelmä:



- Edellisten lisäksi tarvitaan opetus- ja testidataa
- Esikäsittelyn tarkoituksena on korostaa oleellista mittausinformaatiota ja helpottaa piirreirrotusta

- Piirteet ja niiden väliset suhteet ovat hahmon representaatio, jonka perusteella hahmo voidaan luokitella tai kuvata
- Hahmon representaatio voi olla esim. vektori, matriisi, puu, graafi, merkkijono
- Luokitin voi perustua erilaisiin laskennallisiin tekniikoihin (tilastolliset ja geometriset menetelmät, neuroverkot, jne...)
- Kysymykset joihin etsitään vastauksia: "Mitkä hahmoista ovat keskenään samankaltaisia?", "Kuinka monta ja millaisia ryhmiä hahmot muodostavat?"

Kaaviokuva suunnittelusta:



Tunnistusjärjestelmän suunnittelu on iteratiivinen prosessi, jossa keskeisiä kysymyksiä ovat:

- Mitä mitataan ja millä laitteilla?
- Millaisia piirteitä mittauksista voidaan laskea, mitkä piirteet kuvaavat hyvin hahmoja?
- Mitkä piirteistä ovat tehtävän kannalta oleellisia? Mitkä ovat parhaita? Kuinka monta tarvitaan?
- Millainen luokittelutekniikka soveltuu tehtävään parhaiten?
- Kuinka hyvin luokitin toimii opetus- ja testidatalla, kuinka yleisiä saadut tulokset ovat?

3.4 Esikäsittely ja normalisointi

- Onnistunut esikäsittely helpottaa piirreirrotusta ja parantaa luokittelumenetelmän toimivuutta: parempia piirteitä, nopeampi oppiminen, yleisempiä tuloksia
- Esikäsittely- ja normalisointimenetelmien valinta riippuu siitä, millaisia piirteitä halutaan laskea ja mitä luokittelutekniikoita käytetään
- Menetelmien valinta riippuu erittäin paljon sovelluksesta, erilaisia menetelmiä löytyy mm digitaalisen signaalin- ja kuvankäsittelyn piiristä
- Tyypillisiä menetelmiä: ns. outlier ja puuttuvien mittausten käsittely (hylkäys, heuristinen tai tilastollinen päättely), varianssien ja odotustusarvojen normalisointi, pääkomponenttianalyysi, arvoalueen skaalaus (esim. lineaarinen, softmax)

- Softmax-skaalaus:

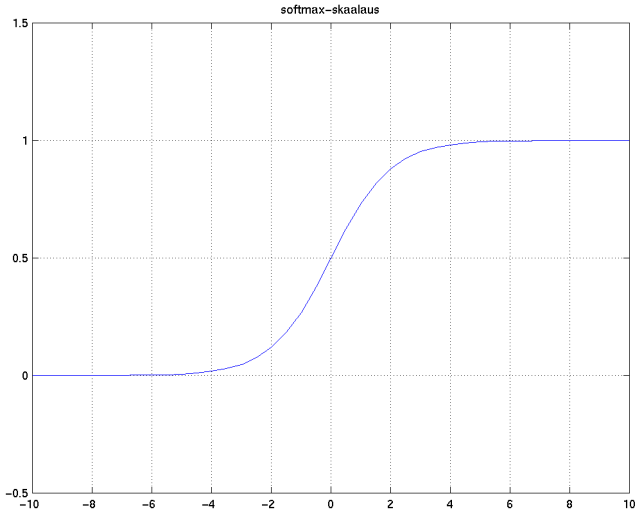
$$\bar{x}_k = \frac{1}{N}\sum_{i=1}^{N} x_{ik}, k = 1, 2, \ldots, l$$

$$\sigma_k^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_{ik} - \bar{x}_k)^2$$

$$y = \frac{x_{ik} - \bar{x}_k}{r\sigma_k}$$

$$\hat{x}_{ik} = \frac{1}{1 + \exp(-y)}$$

(k.:n piirteen käsittely, piirteitä yhteensä l kappaletta) Skaalaus rajoittaa arvot välille $[0, 1]$. Parametri r säätelee aluetta, jossa skaalaus on lineaarinen.



- Pääkomponenttianalyysi käsitellään tarkemmin laskareissa

- Erilaisia hahmojen vääristymiä:

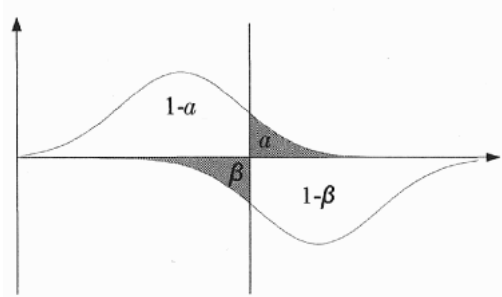
Figure 4: Examples of pattern distortions (visual patterns).
(a) Geometric distortions of visual (image) patterns-square.
(b) Geometric distortions of grid pattern
(c) Geometric distortions of character patterns
(d) More extreme pattern distortions (missing parts and extra parts) using the patterns of parts (a) to (c).

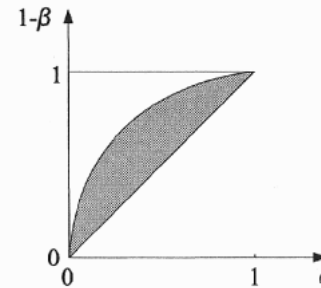
3.5 Piirrevalinta

- Piirteet valitaan siten, että ne kuvaavat hyvin tarkasteltavia hahmoja ja saavat *samankaltaisia arvoja luokkien sisällä* ja *erilaisia arvoja luokkien välillä*
- Piirteet voivat olla symbolisia tai numeerisia, jatkuva-, diskreetti- tai binaariarvoisia
- Piirteet pyritään valitsemaan siten että ne ovat invariantteja erilaisille vääristymille ja mittausolosuhteiden muutoksille (esim. skaalaus, rotaatio, translaatio, valaistus)
- Piirteiden valinta riippuu sovellutuskohteessa!
- Piirteiden valinnalla on ratkaiseva vaikutus luokittelun onnistumiseen
- Hyvien piirteiden valintana sopivia menetelmiä: tilastollinen testaus, "receiver operating characteristic (ROC) curve", erilaiset luokkien separoituvuusmitat

ROC-käyrä:

- Kertoo kuinka päällekkäisiä luokkien jakaumat ovat yhden piirteen suhteen
- Oletus: hahmo kuuluu luokkaan ω_1, jos piirre $x < \theta$, tai, luokkaan ω_2, jos $x \geq \theta$, kokeillaan θ:lle erilaisia arvoja
- α (β) on tn, että luokitellaan hahmo virheellisesti luokkaan ω_2 (ω_1)
- Kun jakaumat ovat täysin päällekkäiset, $\alpha = 1 - \beta$
- Mitä vähemmän päällekkäisyyttä, sitä suurempi alue ROC-käyrän ja edellä mainitun suoran välillä


1-a
a
β
1-β


1-β
1
0
0
1
a

"Ambiguity function":

- Useamman kuin kahden luokan separoituvuusmitta laskettuna useamman kuin yhden piirteen suhteen
- Jaetaan piirteen arvoalue K:ksi intervalleiksi Δ_j, $P(\Delta_j)$ on tn, että kyseiseltä intervallilta on havaintoja, $P(\omega_i|\Delta_j)$ on luokan ehdollinen tn kyseisessä intervallissa, luokkia M kappaletta

$$A = -\sum_{i=1}^{M}\sum_{j=1}^{K} P(\Delta_j)P(\omega_i|\Delta_j)\log_M(P(\omega_i|\Delta_j))$$

- Kun luokkien jakaumat ovat täysin päällekkäiset, $A = 1$, kun jakaumat ovat täysin erilliset, $A = 0$
- Tarvittavat tn:t on helppo estimoida datasta
- Todistus laskareissa

Piirrejoukon valinta

Luokkien separoituvuutta eri piirrevalinnoilla voidaan siis mitata, mutta kuinka valita paras l:n piirteen osajoukko m:stä piirteestä?

- Vaihtoehtoja on runsaasti, kun l:kään ei tunneta:

$$\sum_{i=1}^{m} \binom{m}{i}$$

  eli kaikkia vaihtoehtoja ei yleensä voida evaluoida

- Eräs piirteiden välisiin korrelaatioihin perustuva tekniikka:
  - Laske piirteiden väliset korrelaatiot:

$$\rho_{ij} = \frac{\sum_{n=1}^{N} x_{ni} x_{nj}}{\sqrt{\sum_{n=1}^{N} x_{ni}^2 \sum_{n=1}^{N} x_{nj}^2}}$$

  - Valitse luokkien separoituvuusmitta C
  - Laske C kaikkille piirteille erikseen, järjestä paremmuusjärjestykseen, valitse parhaan C:n tuottama piirre x_{i_1}
  - Valitse seuraava piirre x_{i_2} siten että

$$i_2 = \arg\max_j \{\alpha_1 C(j) - \alpha_2 |\rho_{i_1 j}|\},\ j \neq i_1$$

    α_1 ja α_2 ovat 'hihavakiota'

– Valitse x_{i_k}, $k = 1, \ldots, l$ siten että

$$i_k = \arg\max_j \{\alpha_1 C(j) - \frac{\alpha_2}{k-1} \sum_{r=1}^{k-1} |\rho_{i_r j}|\}$$
$$j \neq i_r,\ r = 1, \ldots, k-1$$

– Variaatio: käytetään kahta separoituvuusmittaa. Silloin

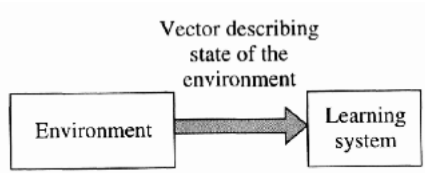
$$i_k = \arg\max_j \{\alpha_1 C_1(j) + \alpha_2 C_2(j) - \frac{\alpha_3}{k-1} \sum_{r=1}^{k-1} |\rho_{i_r j}|\}$$
$$j \neq i_r,\ r = 1, \ldots, k-1$$

3.6 Tunnistusmenetelmät

Tunnistusmenetelmät voidaan jakaa erilaisiin ryhmiin. Yksi tärkeä jakoperuste on oppimisperiaate:

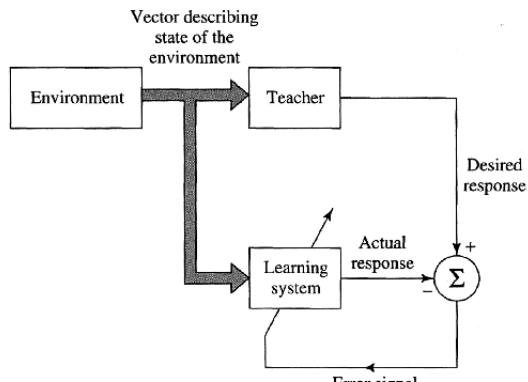
Ohjaamaton oppiminen

- Hahmojen *luokkia ei tiedetä* etukäteen
- Tavoitteena on muodostaa hahmoista ryhmiä, joiden sisällä hahmot ovat samankaltaisia ja joiden välillä on selkeitä eroja (klusterointi)
- Optimoitava funktio on klusteroinnin onnistumista kuvaava mitta
- Aina ei tiedetä edes ryhmien lukumäärää



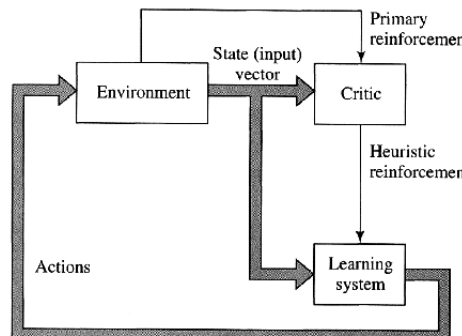
Ohjattu oppiminen

- Hahmojen *luokat tunnetaan* etukäteen
- Tavoitteena on muodostaa kuvaus piirreavaruudesta luokka-avaruuteen
- Optimoitava funktio perustuu kuvauksessa tapahtuviin virheisiin



Vahvistusoppiminen

- "Reinforcement learning"
- Ohjatun ja ohjaamattoman oppimisen välimuoto: luokkia ei tunneta, mutta onnistumisesta saadaan positiivista ja epäonnistumisesta negatiivista palautetta



Jaottelu voi perustua myös käytettyihin laskennallisiin tekniikoihin

Geometriset menetelmät

- Piirreavaruus jaetaan joko lineaarisesti tai epälineaarisesti osiin, jokainen osa vastaa tiettyä luokkaa
- Esim. hypertasot, Fisherin diskriminantti, SVM

Neuroverkkomenetelmät

- Ns. "black box"-menetelmä, joka tekee epälineaarisen kuvauksen piirreavaruudesta luokka-avaruuteen
- Järjestelmän tulkinta hankalaa
- Esim. MLP, RBF, SOM

Tilastolliset menetelmät

- Piirteitä käsitellään tilastollisina muuttujina
- Tunnistus perustuu riskin odotusarvon minimointiin
- Menetelmissä tarvittavien tnjakaumien estimointi voi olla hankalaa

Syntaktiset menetelmät

- Hahmot esitetään esim. formaalin kielen tuottamina merkkijonoina
- Tunnistus perustuu jäsentämiseen
- Jos jäsennys ei onnistu, on vaikea erotella hyviä ja huonoja vastauksia

Rakenteelliset menetelmät

- Hahmot esitetään esim. graafien avulla
- Tunnistus perustuu graafien vertailuun
- Suurien graafien vertailu laskennallisesti raskasta

Malleihin perustuvat menetelmät

- Luokat esitetään tyypillisten, mallihahmojen (prototyypit) avulla
- Tuntematonta hahmoa verrataan malleihin ja tunnistus tehdään samankaltasimpien mallien perusteella
- Vertailu laskennallisesti raskasta ja muistinkulutus korkea, jos malleja on runsaasti

3.7 Evaluointi

Tunnistusjärjestelmää evaluoidaan, jotta tiedetään kuinka se suoriutuu tositilanteessa. Lisäksi evaluointi on läheisesti kytketty kaikkiin muihin järjestelmän suunnitteluvaiheesiin.

Virhetn:n estimointi:

- tutkitaan kuinka monta virhettä syntyy testidatalla luokkaa kohden
- Kun oletetaan, että hahmot ovat toisistaan riippumattomia, luokan *a priori* tn on $P(\omega_i)$ ja luokasta ω_i on N_i näytettä, tn että k_i näytettä tunnistetaan virheellisesti on

$$P(k_i \text{ virhettä}) = \binom{N_i}{k_i} P_i^{k_i} (1 - P_i)^{N_i - k_i}$$

- Yksittäisen luokan ω_i todellista virhetn:ttä P_i ei tunneta

- Yksittäiselle luokalle suurimman uskottavuuden (ML) estimaatti virhetn:lle on

$$\hat{P}_i = \frac{k_i}{N_i}$$

- Kun luokkia on M kappaletta, ML- estimaatti kokonaisvirhetn:lle on

$$\hat{P} = \sum_{i=1}^{M} P(\omega_i) \frac{k_i}{N_i}$$

- ML-estimaatti on harhaton ja sen varianssi on

$$\sigma^2_{\hat{P}} = \sum_{i=1}^{M} P^2(\omega_i) \frac{P_i(1-P_i)}{N_i}$$

  eli estimaatti on sitä tarkempi, mitä enemmän luokista on näytteitä

- Todistuksesta tarkemmin laskareissa

- Jos halutaan, että estimaatti poikkeaa korkeintaan ϵ:n verran todellisesta virhetn:stä tietyllä riskillä a, mikä on testidatan vähimmäismäärä N?

$$\text{prob}\{P \geq \hat{P} + \epsilon(N, a)\} \leq a$$

- Kun oletetaan, että $\epsilon(N, a) = \beta P$, voidaan edellisestä yhtälöstä approksimoida ylä- ja alarajat N:lle.

- Tyypillisesti $a = 0.05$ ja $\beta = 0.2$. Silloin

$$N \approx \frac{100}{P}$$

esim. $P = 0.01, N = 10\,000$

- Huomaa, tulos ei riipu luokkien lukumäärästä

- Jos testidatan näytteet eivät ole riippumattomia, testidataa tarvitaan enemmän

Opetus- ja testidata

Mitä enemmän näytteitä käytetään opetuksessa, sitä paremmin tunnistusjärjestelmä pystyy yleistämään, pienellä testijoukolla vaarana "ylioppiminen"

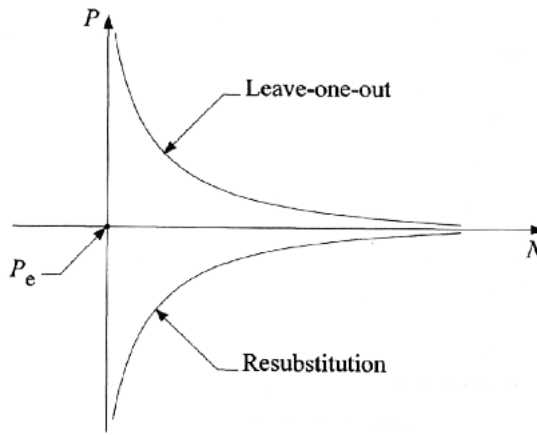
Mitä enemmän näytteitä käytetään evaluointiin, sitä luotettavammat estimaatit virhetn:ille

Rajoitettu määrä dataa, kuinka tehdään jako opetus- ja testidataksi?

Erilaisia lähestymistapoja:

- Käytetään samaa dataa sekä opetukseen että testaukseen ('resubstitution'). Tulokset ylioptimistisia (seuraava kuva)
- Erilliset opetus- ja testidatat. Tulokset realistisempia, sopiva jako löytyy esim. kokeilemalla

- "Leave-One-Out". Opetus tehdään kaikilla paitsi yhdellä näytteellä, jota käytetään testaukseen. Toistetaan kaikilla mahdollisilla osajoukoilla. Laskennallisesti raskasta!
- "Leave-k-Out". Variaatio edellisestä. Ei käydä välttämättä läpi kaikkia osajoukkoja



4. TILASTOLLINEN HAHMONTUNNISTUS

Tilastollisissa hahmontunnistusmenetelmissä piirteitä tarkastellaan tilastollisina muuttujina

Luokittelussa käytetään hyväksi seuraavia tietoja: luokkien *a priori* tn:iä, luokkien ehdollisia tnjakaumia ja tehtyjä havaintoja eli opetusdataa

Hahmon luokka valitaan korkeimman *a posteriori* tn:n tai siitä johdetun funktion mukaisesti tai minimoimalla päätökseen liittyvän riskin odotusarvoa

4.1 Ongelman asettelu

- Hahmo esitetään piirrevektorin $\mathbf{x} \in \mathbb{R}^n$ avulla
- Hahmot halutaan jakaa M:ään luokkaan $\omega_1, \ldots, \omega_M$
- Luokkien *a priori* tn:t ovat $P(\omega_1), \ldots, P(\omega_M)$
- Luokkien ehdolliset tnjakaumat ovat $p(\mathbf{x}|\omega_1), \ldots, p(\mathbf{x}|\omega_M)$

- Jos edellä mainittuja tn:iä ja tnjakaumia ei tunneta, ne voidaan estimoida opetusdatasta:
  - Kun opetusnäytteitä yhteensä N kpl ja luokasta ω_i niitä on N_i kappaletta, $\hat{P}(\omega_i) = N_i/N$
  - Luokkien ehdollisten tnjakaumien estimointia käsitellään myöhemmin
- Mitkä ovat luokkien *a posteriori* tnjakaumat?
- Ol., että jokaisen luokittelutapahtumaan voidaan liittää kustannus: Mitkä ovat eri päätösten riskien odotusarvot?
- Jaetaan piirreavaruus osiin $R_1, \ldots, R_M$ ja tehdään luokittelupäätös seuraavasti: jos $\mathbf{x} \in R_i$, valitaan luokka ω_i
- Mitkä ovat eri luokkia vastaavat alueet piirreavaruudessa, mitkä ovat aluejaon kriteerit?

4.2 "Bayes Decision Theory"

- Optimaalinen tapa suorittaa luokittelu
- Tarkastellaan aluksi kahden luokan, ω_1 ja ω_2, tapausta
- Käyttäen Bayes-sääntöä (Bayes rule) *a posteriori* tn:t saadaan laskettua seuraavasti:

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

missä

$$p(\mathbf{x}) = \sum_{i=1}^{2} p(\mathbf{x}|\omega_i)P(\omega_i) \tag{2}$$

- Jaetaan piirreavaruus osiin R_1 ja R_2 siten, että valitaan aina se luokka, jonka *a posteriori* tn on korkeampi. Luokittelu tehdään siis seuraavasti:

$$\begin{array}{l} \text{Jos } P(\omega_1|\mathbf{x}) > P(\omega_2|\mathbf{x}),\ \mathbf{x} \text{ kuuluu luokkaan } \omega_1 \\ \text{Jos } P(\omega_1|\mathbf{x}) < P(\omega_2|\mathbf{x}),\ \mathbf{x} \text{ kuuluu luokkaan } \omega_2 \end{array} \tag{3}$$

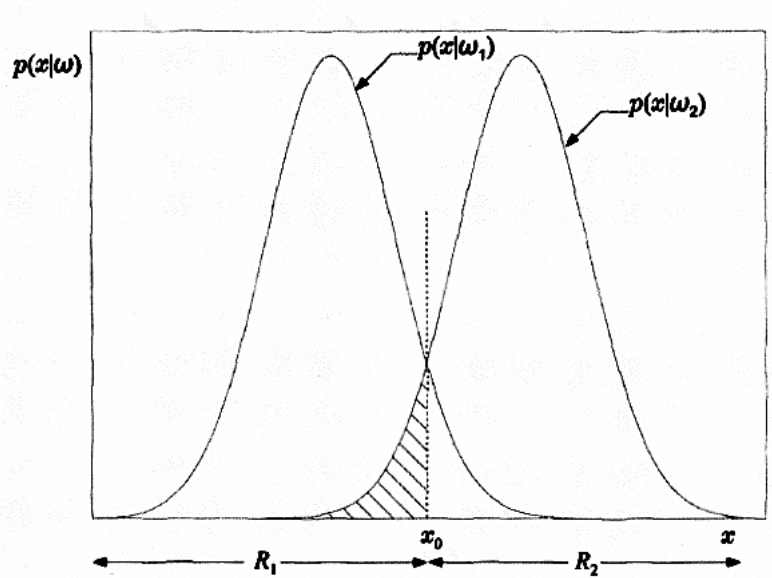
- Edellinen sääntö voidaan kirjoittaa käyttäen kaavaa (1) ja jättämällä irrelevantti $p(\mathbf{x})$ pois myös näin:

$$p(\mathbf{x}|\omega_1)P(\omega_1) \gtrless p(\mathbf{x}|\omega_2)P(\omega_2) \tag{4}$$

- Jos luokkien *a priori* tn:t ovat samat, saadaan:

$$p(\mathbf{x}|\omega_1) \gtrless p(\mathbf{x}|\omega_2) \tag{5}$$

- Kohtaa, jossa luokkien *a posteriori* tn:t ovat samat, kutsutaan päätösrajapinnaksi ja se jakaa piirreavaruuden alueiksi R_1 ja R_2


$p(x|\omega)$
$p(x|\omega_1)$
$p(x|\omega_2)$
x_0
x
R_1
R_2

Luokitteluvirheen tn:n minimointi

- Täydellisen, virheettömän luokittelun saavuttaminen ei ole aina edes teoriassa mahdollista
- Luokitteluvirheen tn voidaan laskea seuraavasti:

$$P_e = P(\mathbf{x} \in R_2, \omega_1) + P(\mathbf{x} \in R_1, \omega_2) \tag{6}$$

- Edellinen kaava voidaan kirjoittaa myös näin:

$$\begin{aligned} P_e &= P(\mathbf{x} \in R_2|\omega_1)P(\omega_1) + P(\mathbf{x} \in R_1|\omega_2)P(\omega_2) \\ &= P(\omega_1) \int_{R_2} p(\mathbf{x}|\omega_1)\, d\mathbf{x} + P(\omega_2) \int_{R_1} p(\mathbf{x}|\omega_2)\, d\mathbf{x} \\ &= \int_{R_2} P(\omega_1|\mathbf{x})p(\mathbf{x})\, d\mathbf{x} + \int_{R_1} P(\omega_2|\mathbf{x})p(\mathbf{x})\, d\mathbf{x} \end{aligned} \tag{7}$$

- Koska R_1:n ja R_2:n unioni kattaa koko piirreavaruuden,

$$\int_{R_1} P(\omega_1|\mathbf{x})p(\mathbf{x})\, d\mathbf{x} + \int_{R_2} P(\omega_1|\mathbf{x})p(\mathbf{x})\, d\mathbf{x} = P(\omega_1) \tag{8}$$

- Yhdistämällä kaavat (7) ja (8) saadaan:

$$P_e = P(\omega_1) - \int_{R_1} (P(\omega_1|\mathbf{x}) - P(\omega_2|\mathbf{x}))p(\mathbf{x})\, d\mathbf{x} \tag{9}$$

- Edellisestä nähdään helposti, että P_e minimoituu silloin, kun

$$\begin{aligned} R_1 &: \ P(\omega_1|\mathbf{x}) > P(\omega_2|\mathbf{x}) \\ R_2 &: \ P(\omega_2|\mathbf{x}) > P(\omega_1|\mathbf{x}), \end{aligned} \tag{10}$$

- Useamman kuin kahden luokan tapaukselle voidaan johtaa vastaavalla päättelyllä luokkiteluvirhetn:n minimoiva päätössääntö:

$$R_i : \ P(\omega_i|\mathbf{x}) > P(\omega_j|\mathbf{x}) \ \forall j \neq i \tag{11}$$

Riskin odotusarvon minimointi

- Luokitteluvirhetn minimointi ei ole paras suunnittelukiriteeri, silloin jos erilaisiin luokittelupäätöksiin liittyy erilaiset riskit,
  esim. "Onko palohälytys oikea vai pelkkä testi?"

- Liitetään kaikkiin luokittelutapahtumiin ($\mathbf{x} \in R_i$, oikea luokka ω_k) kustannuskertoimet λ_{ki}, jotka voidaan koota matriisiksi $L(k,i) = \lambda_{ki}$

- Luokkaan ω_k liittyvä riski:

$$r_k = \sum_{i=1}^{M} \lambda_{ki} \int_{R_i} p(\mathbf{x}|\omega_k)\, d\mathbf{x}, \tag{12}$$

  missä M on luokkien lukumäärä

- Kuinka valita piirreavaruuden jako siten, että riskin odotusarvo r minimoituu?

$$\begin{aligned} r &= \sum_{k=1}^{M} r_k P(\omega_k) \\ &= \sum_{i=1}^{M} \int_{R_i} (\sum_{k=1}^{M} \lambda_{ki} p(\mathbf{x}|\omega_k) P(\omega_k)) d\mathbf{x} \end{aligned} \tag{13}$$

- r minimoituu seuraavalla jaolla:

$$\begin{aligned} &\mathbf{x} \in R_i, \text{ jos } l_i < l_j \ \forall j \neq i \\ &l_m = \sum_{k=1}^{M} \lambda_{km} p(\mathbf{x}|\omega_k) P(\omega_k) \end{aligned} \tag{14}$$

- Huom! Jos valitaan $\lambda_{ki} = 1 - \delta_{ki}$ (δ_{ki} on Kroneckerin delta-funktio), minimoidaan luokitteluvirhetn:ttä

- Kahden luokan tapaus:
  - Eri päätöksiin liittyvät kustannusten odotusarvot:

$$\begin{aligned} l_1 &= \lambda_{11} p(\mathbf{x}|\omega_1) P(\omega_1) + \lambda_{21} p(\mathbf{x}|\omega_2) P(\omega_2) \\ l_2 &= \lambda_{12} p(\mathbf{x}|\omega_1) P(\omega_1) + \lambda_{22} p(\mathbf{x}|\omega_2) P(\omega_2) \end{aligned} \tag{15}$$

  - Valitaan luokka ω_1, kun $l_1 < l_2$:

$$(\lambda_{21} - \lambda_{22}) p(\mathbf{x}|\omega_2) P(\omega_2) < (\lambda_{12} - \lambda_{11}) p(\mathbf{x}|\omega_1) P(\omega_1) \tag{16}$$

  - Yleensä $\lambda_{ij} \geq \lambda_{ii}$. Silloin:

$$\begin{aligned} R_1 &: \frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} > \frac{P(\omega_2)}{P(\omega_1)} \frac{\lambda_{21} - \lambda_{22}}{\lambda_{12} - \lambda_{11}} \\ R_2 &: \frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} < \frac{P(\omega_2)}{P(\omega_1)} \frac{\lambda_{21} - \lambda_{22}}{\lambda_{12} - \lambda_{11}} \end{aligned} \tag{17}$$

    (*a posteriori* tnjakaumien suhde on ns. "likelihood ratio")

- **Esimerkki 1:**
  - Tarkastellaan kahden luokan, ω_1 ja ω_2, ongelmaa ja oletetaan, että $p(x|\omega_1) \sim \mathsf{N}(0, 1/2)$ ja $p(x|\omega_2) \sim \mathsf{N}(1, 1/2)$ eli

$$p(x|\omega_1) = \frac{1}{\sqrt{\pi}} \exp(-x^2)$$
$$p(x|\omega_2) = \frac{1}{\sqrt{\pi}} \exp(-(x-1)^2)$$

    ja että luokkien *a priori* tn:t $P(\omega_1)$ ja $P(\omega_2)$ ovat samat
  - Silloin luokitteluvirhetn:n minimoiva päätösraja on $x_0 : \ \exp(-x^2) = \exp(-(x-1)^2)$ eli $x_0 = 1/2$ (katso kaava (11))

– Kun käytetään seuraava kustannusmatriisia L:

$$L = \begin{bmatrix} 0 & 0.5 \\ 1.0 & 0 \end{bmatrix},$$

$x_0 : \ \exp(-x^2) = 2\exp(-(x-1)^2)$ eli $x_0 = (1 - \ln(2)/2) < 1/2$ (katso kaava (16))

– Huom! Jos luokkien *a priori* tn:t eivät ole samat, $P(\omega_1) \lessgtr P(\omega_2)$, siirtyy päätösraja myöskin vasemmalle tai oikealle

- **Esimerkki 2:**

  – Palohälytyksen luokittelu eli juostaanko ulos (tulipalo) vai jäädäänkö sisälle ihmettelemään (väärä hälytys)?

  – Olkoot $\omega_1 =$ "tulipalo", $\omega_2 =$ "väärä hälytys"

  – Oletetaan *a priori* tn:n sille että talossa on tulipalo olevan 1 päivä 10:ssä vuodessa eli $P(\omega_1) = 1/3650$ ja $P(\omega_2) = 3649/3650$

- Piirrevektori $\mathbf{x}$ koostuu esim. seuraavista havainnoista: kuinka iso osa muista ihmisistä säntää ulos, kuinka sakeaa on savu, näkyykö liekkejä jne
- $p(\mathbf{x}|\omega_1)$ ja $p(\mathbf{x}|\omega_2)$ arvioidaan aikaisempien kokemusten pohjalta
- Liitetään eri luokittelupäätöksiin seuraavat ajassa mitatut kustannukset: $\lambda_{11} =$ "kymmenen vuoden tulot ja 1 tunti pihalla", $\lambda_{12} =$ "kymmenen vuoden tulot ja 60 vuotta loppu elämästä", $\lambda_{21} =$ "1 tunti pihalla", $\lambda_{22} = 0$
- Kaavan (16) perusteella riskin odotusarvo minimoituu,kun päätös tehdään seuraavasti:

$$\text{tulipalo}: \ \frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} > \frac{3649}{525599} \approx 0.007$$

$$\text{väärä hälytys}: \ \text{muulloin}$$

  eli pelkkien havaintojen perusteella pitää olla reilusti yli 100-kertainen luottamus siihen ettei ole tulipaloa, jos aikoo jäädä sisälle

4.3 Diskriminanttifunktiot ja päätöspinnat

- Edellisten tarkastelujen pohjalta tiedetään, että kun luokittelu perustuu joko luokitteluvirhetn:n tai riskin odotusarvon minimointiin, piirreavaruus jaetaan M:ään **päätösalueeseen** $R_1, \ldots, R_M$, kun luokkia on M kappaletta

- Mikäli luokkia ω_i ja ω_j vastaavat, luokitteluvirheen tn:n minimoivat päätösalueet ovat R_i ja R_j, määritellään **päätöspinta** (decision boundary, decision surface) seuraavasti:

$$P(\omega_i|\mathbf{x}) - P(\omega_j|\mathbf{x}) = 0 \tag{18}$$

  Toisella puolella päätöspintaa erotus on positiivinen ja toisella negatiivinen

- Toisinaan on laskennallisesti kätevämpää esittää päätöspinnan yhtälö **diskriminanttifunktioiden** $g_i(\mathbf{x}) = f(P(\omega_i|\mathbf{x}))$ avulla. $f(\cdot)$ voi olla mikä tahansa monotonisesti kasvava, jatkuva funktio

- Tällöin minivirhetn:n tuottama päätössääntö (11) saa seuraavan muodon:

$$R_i : \ g_i(\mathbf{x}) > g_j(\mathbf{x}) \ \forall j \neq i \tag{19}$$

ja päätöspinnat on määritelty seuraavasti:

$$g_{ij}(\mathbf{x}) = g_i(\mathbf{x}) - g_j(\mathbf{x}) = 0 \tag{20}$$

4.4 Bayesiläinen luokitin normaalijakaumille

Millaisia päätöspintoja syntyy, kun luokat ovat normaalijakautuneita?

- l-uloitteisen, luokkaan ω_i kuuluvan piirrevektorin $\mathbf{x}$ normaalijakauma:

$$p(\mathbf{x}|\omega_i) = \frac{1}{(2\pi)^{l/2}|\mathbf{\Sigma}_i|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}-\mu_i)^T\mathbf{\Sigma}_i^{-1}(\mathbf{x}-\mu_i)), \quad (21)$$

  missä μ_i ja $\mathbf{\Sigma}_i$ ovat luokan ω_i odotusarvo ja kovarianssimatriisi

$$\mu_i = \mathsf{E}[\mathbf{x}] \quad (22)$$
$$\mathbf{\Sigma}_i = \mathsf{E}[(\mathbf{x}-\mu_i)(\mathbf{x}-\mu_i)^T] \quad (23)$$

- Päätöspinnan yhtälöstä tulee kätevämpi, kun lasketaan diskriminanttifunktiot käyttäen $\ln(\cdot)$-funktiota:

$$\begin{aligned} g_i(\mathbf{x}) &= \ln(\frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})}) \\ &= \ln(p(\mathbf{x}|\omega)i)) + \ln(P(\omega_i)) - \ln(p(\mathbf{x})) \end{aligned} \quad (24)$$

- Sijoittamalla kaava (21) edelliseen kaavaan saadaan:

$$\begin{aligned} g_i(\mathbf{x}) &= -\frac{1}{2}(\mathbf{x}-\mu_i)^T\mathbf{\Sigma}_i^{-1}(\mathbf{x}-\mu_i) + \ln(P(\omega_i)) + c_i \\ c_i &= -\frac{l}{2}\ln(2\pi) - \frac{1}{2}\ln(|\mathbf{\Sigma}_i|) + \ln(p(\mathbf{x})) \end{aligned} \tag{25}$$

- Huom! päätöspinnat $g_{ij}(\mathbf{x})$ ovat kvadraattisia

- **erikoistapauksia:** Seuraavissa tarkasteluissa oletetaan, että luokkien kovarianssimatriisit ovat samat eli $\mathbf{\Sigma}_i = \mathbf{\Sigma}\ \forall i = 1, \ldots, M$

  Silloin kaavan (25) kvadraattinen osuus $\mathbf{x}^T\mathbf{\Sigma}_i^{-1}\mathbf{x}$ ja vakiotermit ovat kaikille luokille samat ja ne voidaan jättää huomioimatta

  Diskrimanantifunktiot voidaan esittää seuraavasti:

$$\begin{aligned} &g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_{i0} \\ &\mathbf{w}_i = \mathbf{\Sigma}^{-1}\mu_i \\ &w_{i0} = \ln(P(\omega_i)) - \frac{1}{2}\mu_i^T\mathbf{\Sigma}^{-1}\mu_i \end{aligned} \tag{26}$$

eli päätöspinnat ovat lineaarisia hypertasoja

Tapaus 1: Piirteet ovat toisistaan riippumattomia, piirteiden varianssit ovat samat:

- Kovarianssimatriisi $\mathbf{\Sigma}$ on muotoa $\mathbf{\Sigma} = \sigma^2\mathbf{I}$
- Diskriminanttifunktiot ovat muotoa:

$$g_i(\mathbf{x}) = \frac{1}{\sigma^2}\mu_i^T\mathbf{x} + w_{i0} \tag{27}$$

- Päätöspinnat ovat muotoa:

$$\begin{aligned} g_{ij}(\mathbf{x}) &= g_i(\mathbf{x}) - g_j(\mathbf{x}) = \mathbf{w}^T(\mathbf{x} - \mathbf{x}_0) \\ \mathbf{w} &= \mu_i - \mu_j \\ \mathbf{x}_0 &= \frac{1}{2}(\mu_i + \mu_j) - \sigma^2 \ln(\frac{P(\omega_i)}{P(\omega_j)})\frac{\mu_i - \mu_j}{\|\mu_i - \mu_j\|^2} \end{aligned} \tag{28}$$

- Geometrinen tulkinta: päätöspinta on pisteen $\mathbf{x}_0$ kautta kulkeva suora, joka on kohtisuora vektoriin $\mu_i - \mu_j$ nähden. Pisteen $\mathbf{x}_0$ sijainti riippuu luokkien variansseista ja *a priori* tn:istä

– Yhteys minimietäisyys luokittimeen: Jos luokkien *a priori* tn:t ovat samat, saadaan vastaava luokitin, kun luokitellaan $\mathbf{x}$ siihen luokkaan, jonka odotusarvo on lähempänä Euklidisen metriikan mukaan. Etäisyyden tasa-arvokäyrät ovat ympyröitä

Tapaus 2: Piirteet ovat riippuvaisia toisistaan

– Päätöspinnat ovat muotoa:

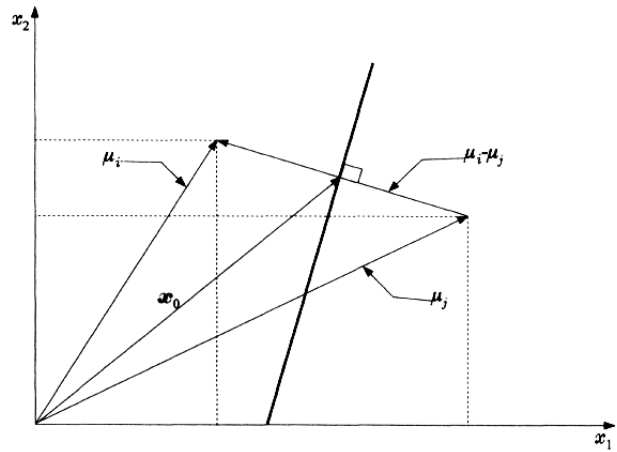
$$\begin{aligned} g_{ij}(\mathbf{x}) &= g_i(\mathbf{x}) - g_j(\mathbf{x}) = \mathbf{w}^T(\mathbf{x} - \mathbf{x}_0) \\ \mathbf{w} &= \mathbf{\Sigma}^{-1}(\mu_i - \mu_j) \\ \mathbf{x}_0 &= \frac{1}{2}(\mu_i + \mu_j) - \ln(\frac{P(\omega_i)}{P(\omega_j)})\frac{\mu_i - \mu_j}{\|\mu_i - \mu_j\|^2_{\mathbf{\Sigma}^{-1}}}, \end{aligned} \tag{29}$$

missä $\|\mathbf{x}\| = (\mathbf{x}^T\mathbf{\Sigma}^{-1}\mathbf{x})^{1/2}$ on $\mathbf{x}$:n ns $\mathbf{\Sigma}^{-1}$-normi eli Mahalanobis-etäisyys origosta

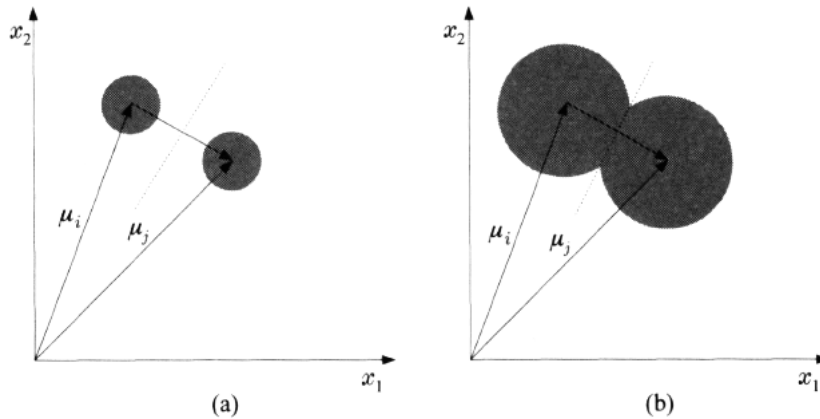
– Geometrinen tulkinta: päätöspinta on pisteen $\mathbf{x}_0$ kautta kulkeva suora, joka on kohtisuora vektoriin $\mathbf{\Sigma}^{-1}(\mu_i - \mu_j)$ nähden

– Yhteys minimietäisyys luokittimeen: Jos luokkien *a priori* tn:t ovat samat, saadaan vastaava luokitin, kun luokitellaan $\mathbf{x}$ siihen luokkaan, jonka odotusarvo on lähempänä Mahalanobis-metriikan mukaan. Etäisyyden tasa-arvokäyrät ovat ellipsejä, joiden akselien suhde saadaan $\mathbf{\Sigma}^{-1}$:n ominaisarvoista

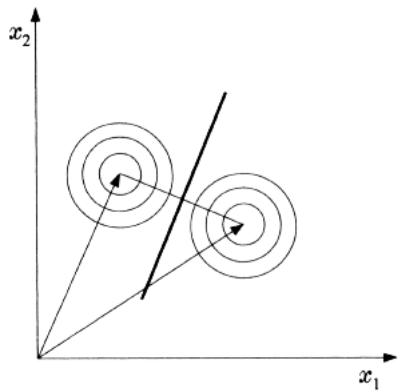
Tapaus 1: Päätöspinnan sijoittuminen

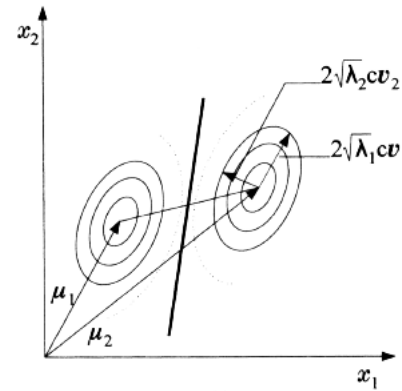


Tapaus 1: Luokkien varianssien ja *a priori* tn:ien vaikutus päätöspinnan sijoittumiseen



Tapaukset 1 ja 2: Yhteys minimietäisyysluokittimeen





5. TODENNÄKÖISYYSJAKAUMIEN ESTIMOINTI

Edellä esitelty Bayesiläinen luokittelusääntö ('Bayes Decision Theory') on optimaalinen tapa suorittaa luokittelu, kun luokkien tnjakaumat tunnetaan

Käytännössä tnjakaumia ei tunneta, vaan ne on estimoitava havainnoista

Voi olla, että tunnetaan tnjakauman tyyppi (esim. normaalijakauma), mutta ei parametrejä (esim. odotusarvo, varianssi); tilanne voi olla myös täysin päinvastainen

Seuraavaksi käydään läpi menetelmiä, joiden avulla voidaan estimoida tilastollisessa tunnistuksessa tarvittavia tnjakaumia

5.1 Suurimman uskottavuuden menetelmä

'Maximum Likelihood Parameter Estimation', ML-estimaatti

Tarkastellaan M:n luokan tunnistusongelmaa

Tehdään seuraavat oletukset:

- Ol., että tunnetaan tnjakaumien $p(\mathbf{x}|\omega_i)$ tyyppi ja että ne voidaan määrittää parametrivektoreiden $\mathbf{\Theta}_i$ avulla. Merkitään siis $p(\mathbf{x}|\omega_i; \mathbf{\Theta}_i)$
- Ol., että eri luokista tehdyt havainnot eivät riipu toisistaan ja tnjakaumat voidaan estimoida erikseen luokkakohtaisesti
- Ol., että tehdyt havainnot ovat toisistaan riippumattomia myös luokkien sisällä

Ongelma: kuinka valitaan tnjakauman $p(\mathbf{x}|\mathbf{\Theta})$ parametrit $\mathbf{\Theta}$, kun on tehty havainnot $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$?

Vastaus: Maksimoidaan parametrien uskottavuusfunktio ('likelihood function') $p(\mathbf{X}; \mathbf{\Theta})$:

$$p(\mathbf{X}; \mathbf{\Theta}) = p(\mathbf{x}_1, \ldots, \mathbf{x}_N; \mathbf{\Theta}) = \prod_{k=1}^{N} p(\mathbf{x}_k; \mathbf{\Theta}) \tag{30}$$

Eli valitaan estimaatti $\hat{\mathbf{\Theta}}_{ML}$ seuraavasti:

$$\hat{\mathbf{\Theta}}_{ML} = \arg\max_{\mathbf{\Theta}} \prod_{k=1}^{N} p(\mathbf{x}_k; \mathbf{\Theta}) \tag{31}$$

Välttämätön ehto optimiratkaisulle $\hat{\mathbf{\Theta}}_{ML}$:

$$\frac{\partial \prod_{k=1}^{N} p(\mathbf{x}_k; \mathbf{\Theta})}{\partial \mathbf{\Theta}} = \mathbf{0} \tag{32}$$

Voidaan yhtä hyvin tarkastella uskottavuusfunktion logaritmiä ('loglikelihood function') $L(\boldsymbol{\Theta})$:

$$L(\boldsymbol{\Theta}) = \ln(\prod_{k=1}^{N} p(\mathbf{x}_k; \boldsymbol{\Theta})) = \sum_{k=1}^{N} \ln(p(\mathbf{x}_k; \boldsymbol{\Theta})) \tag{33}$$

Silloin ehto (32) saa muodon

$$\begin{aligned} \hat{\boldsymbol{\Theta}}_{ML} : \frac{\partial L(\boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}} &= \sum_{k=1}^{N} \frac{\partial \ln(p(\mathbf{x}_k; \boldsymbol{\Theta}))}{\partial \boldsymbol{\Theta}} \\ &= \sum_{k=1}^{N} \frac{1}{p(\mathbf{x}_k; \boldsymbol{\Theta})} \frac{\partial p(\mathbf{x}_k; \boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}} \\ &= \mathbf{0} \end{aligned} \tag{34}$$

ML-estimaatin ominaisuuksia:

- ML-estimaatti on *asymptoottisesti harhaton* ts sen odotusarvo konvergoi kohti parametrivektorin todellista arvoa $\mathbf{\Theta}_0$, kun havaintojen lukumäärä lähestyy ääretöntä ('convergence in the mean'):

$$\lim_{N\to\infty} \mathsf{E}[\hat{\mathbf{\Theta}}_{ML}] = \mathbf{\Theta}_0 \tag{35}$$

- ML-estimaatti on *asymptoottisesti konsistentti* ts kun havaintoja on paljon, sitä todennäköisemmin estimaatti poikkeaa todellisesta arvosta mielivaltaisen vähän. ML-estimaatti toteuttaa seuraavat yhtälöt:

$$\lim_{N\to\infty} \Pr(\|\hat{\mathbf{\Theta}}_{ML} - \mathbf{\Theta}_0\| \leq \epsilon) = 1 \tag{36}$$

('convergence in probability')

$$\lim_{N\to\infty} \mathsf{E}[\|\hat{\mathbf{\Theta}}_{ML} - \mathbf{\Theta}_0\|^2] = 0 \tag{37}$$

('convergence in the mean square')

- ML-estimaatti on *asymptoottisesti tehokas* eli sen varianssi on Cramer-Rao alarajalla
- ML-estimaatin tnjakauma lähestyy normaalijakaumaa, kun havaintojen lukumäärä lähestyy ääretöntä (keskeinen raja-arvolause!)

Esimerkki: normaalijakauman odotusarvon ML-estimaatti

Ol., että havainnot $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ ovat peräisin normaalijakaumasta $p(\mathbf{x}_k; \mu, \mathbf{\Sigma})$, jonka odotusarvoa ei tunneta

Silloin:

$$p(\mathbf{x}_k; \mu, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{l/2}|\mathbf{\Sigma}|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}_k - \mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}_k - \mu))$$

$$L(\mu) = \ln \prod_{k=1}^{N} p(\mathbf{x}_k; \mu, \mathbf{\Sigma})$$

$$= -\frac{N}{2} \ln((2\pi)^l |\mathbf{\Sigma}|) - \frac{1}{2} \sum_{k=1}^{N} (\mathbf{x}_k - \mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}_k - \mu)$$

Lasketaan gradientti μ:n suhteen ja etsitään nollakohta:

$$\frac{\partial L(\mu)}{\partial \mu} = \sum_{k=1}^{N} \mathbf{\Sigma}^{-1}(\mathbf{x}_k - \mu) = \mathbf{0}$$

$$\hat{\mu}_{ML} = \frac{1}{N} \sum_{k=1}^{N} \mathbf{x}_k$$

5.2 Suurimman a posteriori tn:n menetelmä

'Maximum *A Posteriori* Probability Estimation', MAP-estimaatti

Tarkastellaan M:n luokan tunnistusongelma ja tehdään samanlaiset lähtöoletukset kuin ML-estimaatin tapauksessa

Maksimoidaan parametrien uskottavuusfunktion $p(\mathbf{X};\mathbf{\Theta})$ sijasta parametrien *a posteriori* tn:ttä $p(\mathbf{\Theta}|\mathbf{X})$

Bayes-säännön perusteella:

$$p(\mathbf{\Theta}|\mathbf{X}) = \frac{p(\mathbf{\Theta})p(\mathbf{X}|\mathbf{\Theta})}{p(\mathbf{X})} \tag{38}$$

MAP-estimaatti $\hat{\mathbf{\Theta}}_{MAP}$ löytyy $p(\mathbf{\Theta}|\mathbf{X})$:n (tai sen logaritmin!) maksimikohdasta eli

$$\hat{\mathbf{\Theta}}_{MAP} : \frac{\partial p(\mathbf{\Theta}|\mathbf{X})}{\partial \mathbf{\Theta}} = \frac{\partial p(\mathbf{\Theta})p(\mathbf{X}|\mathbf{\Theta})}{\partial \mathbf{\Theta}} = \mathbf{0} \tag{39}$$

Ero ML- ja MAP-estimaatin välillä on *a priori* tn:n $p(\mathbf{\Theta})$ käytössä. Jos $p(\mathbf{\Theta})$ on tasajakauma, ML- ja MAP-estimaatit ovat samat. Jos $p(\mathbf{\Theta})$:iin liittyvä varianssi on suuri, estimaatit ovat lähestulkoon samat

Esimerkki: normaalijakauman odotusarvon MAP-estimaatti

Ol., että havainnot $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ ovat peräisin normaalijakaumasta $p(\mathbf{x}_k; \mu, \mathbf{\Sigma})$, jonka odotusarvoa ei tunneta

Ol., että piirrevektorien kovarianssimatriisi $\mathbf{\Sigma}$ on muotoa $\mathbf{\Sigma} = \sigma^2\mathbf{I}$

Ol., että normaalijakauman odotusarvo on normaalijakautunut $N(\mu_0, \sigma_\mu^2\mathbf{I})$ eli sen *a priori* tnjakauma on seuraava:

$$p(\mu) = \frac{1}{(2\pi)^{(l/2)}\sigma_\mu^l} \exp(-\frac{1}{2}\frac{\|\mu - \mu_0\|^2}{\sigma_\mu^2})$$

Silloin MAP-estimaatti $\hat{\mu}_{MAP}$ löytyy seuraavalla tavalla:

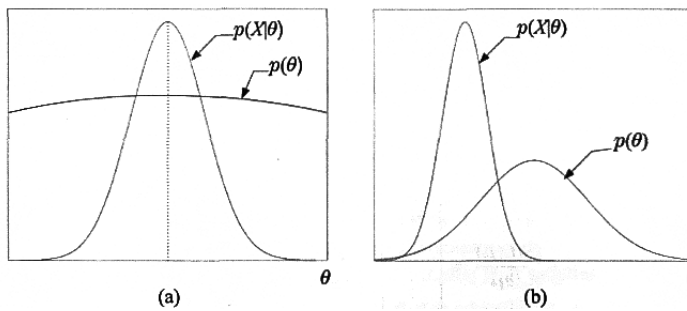
$$\frac{\partial}{\partial \mu} \ln(\prod_{k=1}^{N} p(\mathbf{x}_k|\mu)p(\mu)) = 0$$

$$\sum_{k=1}^{N} \frac{1}{\sigma^2}(\mathbf{x}_k - \hat{\mu}_{MAP}) - \frac{1}{\sigma_\mu^2}(\hat{\mu}_{MAP} - \mu_0) = 0$$

$$\hat{\mu}_{MAP} = \frac{\mu_0 + \frac{\sigma_\mu^2}{\sigma^2} \sum_{k=1}^{N} \mathbf{x}_k}{1 + \frac{\sigma_\mu^2}{\sigma^2} N}$$

Silloin kun $\frac{\sigma_\mu^2}{\sigma^2} \gg 1$ ML- ja MAP-estimaatit ovat lähestulkoon samat

Seuraavassa kuvassa kaksi tapausta, joissa erilaiset uskottavuusfunktiot ja *a priori* tnjakaumat. Tapauksessa a) ML- ja MAP-estimaatit ovat samankaltaiset, tapauksessa b) niiden ero on selvä



5.3 Bayesiläinen päättely

ML- ja MAP-menetelmät löytävät tnjakaumien $p(\mathbf{x}|\omega_i; \mathbf{\Theta}_i)$ parametreille arvot käyttäen hyväksi tehtyjä havaintoja $\mathbf{X}_i$ ja *a priori* tnjakaumia $p(\mathbf{\Theta}_i)$

Miksi emme suoraan laskisi tnjakaumia $p(\mathbf{x}|\mathbf{X_i})$ ja suorittaisi luokittelua näiden avulla?

Tehdään taas samat lähtöoletukset kuin ML- ja MAP-estimaattien tapauksissa

Tiedetään, että

$$p(\mathbf{x}|\mathbf{X}) = \int p(\mathbf{x}|\mathbf{\Theta})p(\mathbf{\Theta}|\mathbf{X})d\mathbf{\Theta}, \tag{40}$$

missä

$$p(\mathbf{\Theta}|\mathbf{X}) = \frac{p(\mathbf{X}|\mathbf{\Theta})p(\mathbf{\Theta})}{p(\mathbf{X})} = \frac{p(\mathbf{X}|\mathbf{\Theta})p(\mathbf{\Theta})}{\int p(\mathbf{X}|\mathbf{\Theta})p(\mathbf{\Theta})d\mathbf{\Theta}} \tag{41}$$

$$p(\mathbf{X}|\mathbf{\Theta}) = \prod_{k=1}^{N} p(\mathbf{x}_k|\mathbf{\Theta}) \tag{42}$$

Tämän menetelmän haittapuoli on sen monimutkaisuus – analyyttinen ratkaisu on olemassa vain erikoistapauksille

Esimerkki: normaalijakauman päättely Bayesiläisittäin

Ol., että $p(x|\mu) = N(\mu, \sigma^2)$, missä μ on tuntematon

Ol., että $p(\mu) = N(\mu_0, \sigma_0^2)$

Kun käytettävissä on N havaintoa, saadaan $p(\mu|X) = N(\mu_N, \sigma_N^2)$, missä

$$\mu_N = \frac{N\sigma_0^2\bar{x} + \sigma\mu_0}{N\sigma_0^2 + \sigma^2}$$
$$\sigma_N = \frac{\sigma^2\sigma_0^2}{N\sigma_0^2 + \sigma^2}$$
$$\bar{x} = \frac{1}{N}\sum_{k=1}^{N} x_k$$

Kun havaintojen määrä lähestyy ääretöntä, Bayesiläisellä päättelyllä saatu jakauma lähestyy piikkiä, jonka huipun kohta on ML-estimaatti. Tämä tulos yleistyy useille muillekin paljon käytetyille tnjakaumille

Huom! Kun N lähestyy ääretöntä, ML- ja MAP-menetelmillä sekä Bayesiläisellä päättelyllä saadut tulokset lähestyvät toisiaan. Menetelmien erot ovat merkittäviä, kun käytettävissä on vain rajallisesti havaintoja

5.4 Maksimientropiamenetelmä

Edellä esitetyissä menetelmissä oletettiin tnjakauman tyyppi tunnetuksi ja estimoitiin sille sopivat parametrit

Entä jos tunnetaan joitain tilastollisia tunnuslukuja, mutta ei tiedetä mikä tnjakaumatyyppi on kyseessä?

Valitaan sellainen tnjakauma, jolla on tunnetut ominaisuudet ja maksimaalinen entropia (ei aina mikään helppo ongelma!)

Tnjakauman $p(\mathbf{x})$ entropia H on määritelty seuraavasti:

$$H = -\int_{\mathbf{x}} p(\mathbf{x}) \ln(p(\mathbf{x})) d\mathbf{x} \tag{43}$$

Entropia kuvaa ilmiön satunnaisuutta tai yllätyksellisyyttä

Tnjakauman entropian maksimointi vastaa minimimaalisinta *a priori* -tiedon käyttöä

Esimerkki: milloin maksimientropiajakauma on tasajakauma?

Ongelma, maksimoi H, kun tiedetään että $x_1 \leq x \leq x_2$, eli

$$\int_{x_1}^{x_2} p(x)dx = 1$$

Lagrangen kertoimien avulla saadaan seuraava ekvivalentti ongelma:

$$\max_{p(x)} H_L = \max_{p(x)} - \int_{x_1}^{x_2} p(x)(\ln(p(x)) - \lambda)dx - \lambda$$

Derivoidaan edellinen $p(x)$:n suhteen ja asetetaan nollaksi:

$$\frac{\partial H_L}{\partial p(x)} = \int_{x_1}^{x_2} [(\ln p(x) - \lambda) + 1]dx = 0$$
$$\hat{p}(x) = \exp(\lambda - 1)$$

Saadun estimaatin pitää toteuttaa rajoitusehto, joten $\exp(\lambda - 1) = \frac{1}{x_2 - x_1}$ ja

$$\hat{p}(x) = \begin{cases} \frac{1}{x_2 - x_1}, & \text{jos } x_1 \leq x \leq x_2 \\ 0, & \text{muulloin} \end{cases}$$

Huom! Voidaan osoittaa, että jos tunnetaan odotusarvo ja varianssi, maksimientropiaratkaisu on normaalijakauma

5.5 Epäparametriset menetelmät

Edellä esitetyissä menetelmissä oletettiin, että tnjakauma voidaan määrittää parametrivektorin avulla. Seuraavaksi tarkastellaan menetelmiä, joissa tätä oletusta ei tehdä

Sekä Parzen-ikkunat ja k:n lähimmän naapurin menetelmä ovat eräänlaisia variaatioita tnjakauman approksimoinnista histogrammin avulla

Tnjakaumaa voidaan approksimoida histogrammin avulla seuraavasti:

- Tarkastellaan vain yhtä piirrettä x , eli 1-ulotteista tapausta, ja pyritään approksimoimaan tnjakauma $p(x)$
- Jaetaan x-akseli h:n pituisiksi intervalleiksi
- Approksimoidaan tn P, että x:n arvo osuu tietylle intervallille. Olkoot N havaintojen lkm ja k_N intervallille osuneiden havaintojen lkm. Silloin $P \approx k_N/N$
- Kun N lähestyy ääretöntä,

$$\hat{p}(x) \equiv \hat{p}(\hat{x}) \approx \frac{1}{h}\frac{k_N}{N},\ |x - \hat{x}| \leq \frac{h}{2}, \tag{44}$$

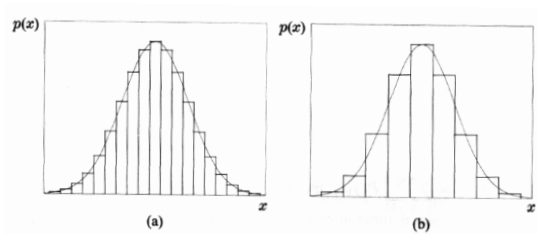
  missä $\hat{x}$ on intervallin keskikohta
- Approksimaatio on hyvä silloin, kun $p(x)$ on jatkuva ja h on riittävän pieni eli oletus $p(x) =$ vakio intervallilla on järkevä

- Voidaan osoittaa, että $\hat{p}(x) \to p(x)$, jos $N \to \infty$ ja seuraavat oletukset toteutuvat:

$$h_N \to 0$$
$$k_N \to \infty$$
$$\frac{k_N}{N} \to 0$$

- Käytännössä N on rajoitettu ja sopiva h pitää valita itse

Seuraavassa kuvassa tnjakaumaa approksimoidaan histogrammin avulla. Tapauksessa a) on valittu h:lle pieni ja tapauksessa b) suuri arvo



Parzen-ikkunat

Moniulotteisessa tapauksessa l-ulotteinen piirreavaruus jaetaan hyperkuutioksi, joiden tilavuus on h^l

Määritellään jokaiselle havainnolle $\mathbf{x}_i$ kantafunktio $\Phi(\mathbf{x}_i)$ seuraavasti:

$$\Phi(\mathbf{x}_i) = \begin{cases} 1, & \text{kun } |x_{ij}| \leq 1/2 \\ 0, & \text{muulloin} \end{cases}, \tag{45}$$

missä x_{ij} on $\mathbf{x}_i$:n j. komponentti

Kaava (44) voidaan esittää tällaisten funktioiden summana:

$$\hat{p}(\mathbf{x}) = \frac{1}{h^l}\left(\frac{1}{N}\sum_{i=1}^{N}\Phi\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)\right) \tag{46}$$

Edellä jatkuvaa tnjakaumaa approksimoitiin epäjatkuvilla kuutiolla. Jatkuva approksimaatio saadaan, kun käytetään jatkuvia kantafunktioita $\Phi(\cdot)$

Kantafunktiot valitaan siten, että

$$\begin{aligned} \Phi(\mathbf{x}) &\geq 0 \\ \int_{\mathbf{x}} \Phi(\mathbf{x}) d\mathbf{x} &= 1 \end{aligned} \tag{47}$$

Kantafunktiot voivat olla esim. eksponentiaalisia, $N(\mathbf{0}, \mathbf{I})$

Mikä on $\hat{p}(\mathbf{x})$:n odotusarvo eli kuinka approksimaatio käyttäytyy, kun $N \to \infty$?

- $\hat{p}(\mathbf{x})$ on määritelty havaintojen $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ avulla, jotka noudattavat todellista tnjakaumaa $p(\mathbf{x})$
- Lasketaan $\hat{p}(\mathbf{x})$:n odotusarvo tnjakauman $p(\mathbf{x})$ suhteen:

$$\mathsf{E}[\hat{p}(\mathbf{x})] = \frac{1}{h^l}(\frac{1}{N}\sum_{i=1}^{N} \mathsf{E}[\Phi(\frac{\mathbf{x}_i - \mathbf{x}}{h})]) = \int_{\xi} \frac{1}{h^l}\Phi(\frac{\xi - \mathbf{x}}{h}) p(\xi) d\xi \tag{48}$$

- Yllä olevasta kaavasta nähdään, että $\hat{p}(\mathbf{x})$ on tasoitettu ('smoothed') versio todellisesta jakaumasta

- Kun $h \to 0$, $\frac{1}{h^l}\Phi(\frac{\xi-\mathbf{x}}{h})$ lähestyy deltafunktiota $\delta(\xi - \mathbf{x})$ ja nähdään, että $\hat{p}(\mathbf{x})$ on harhaton estimaatti $p(\mathbf{x})$:lle riippumatta N:stä

Jos N on kiinnitetty, estimaatin $\hat{p}(\mathbf{x})$ varianssi kasvaa, kun h pienenee

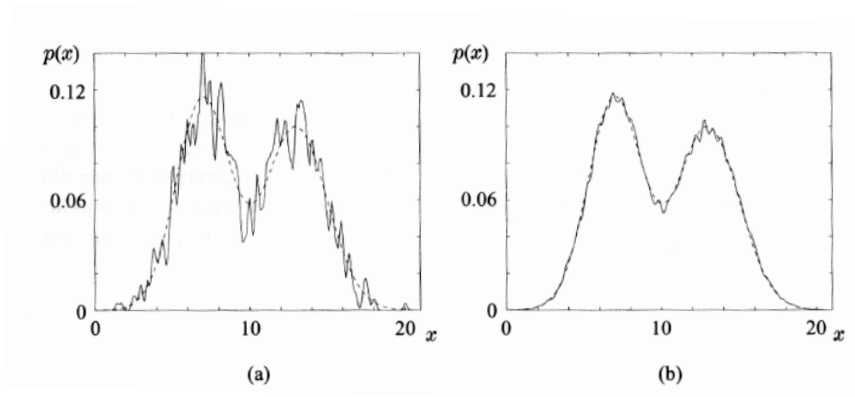
Jos h on kiinnitetty, estimaatin $\hat{p}(\mathbf{x})$ varianssi pienenee, kun $N \to \infty$

Useimmilla kantafunktiolla saatu estimaatti $\hat{p}(\mathbf{x})$ on harhaton ja asymptoottisesti konsistentti, jos $h \to 0$ siten, että $hN \to \infty$, kun $N \to \infty$

Käytännössä N on rajoitettu ja h joudutaan valitsemaan itse, esim. minimoimalla luokitteluvirhettä

Mikäli ei haluta tinkiä estimaatin ominaisuuksista, tarvittavien havaintojen N lukumäärä kasvaa eksponentiaalisesti piirrevektoreiden dimension l suhteen ('curse of dimensionality!')

Seuraavassa kuvassa on esitetty Parzen-ikkunoilla saatuja tnjakauman (katkoviiva) approksimaatioita. Kantafunktiot ovat normaalijakaumia ja $h = 0.1$. Tapauksessa a) $N = 1000$ ja tapauksessa b) $N = 20\,000$



k:n lähimmän naapurin menetelmä

Edellisessä menetelmässä kantafunktiot olivat samanlaisia riippumatta siitä oliko tarkasteltavassa piirreavaruuden osassa havaintoja tiheässä vai harvassa

Yleisempi versio kaavalle (44) :

$$\hat{p}(\mathbf{x}) = \frac{k}{NV(\mathbf{x})}, \tag{49}$$

missä $V(\mathbf{x})$ on $\mathbf{x}$:stä riippuva tilavuus. Eli kiinnitetään ensin k ja lasketaan vasta sitten havaintojen viemä tilavuus

Voidaan osoittaa, että estimaatti on harhaton ja asymptoottisesti konsistentti, jos $k \to \infty$ ja $k/N \to 0$, kun $N \to \infty$

Saatua estimaattia kutsutaan tnjakauman k:n lähimmän naapurin estimaatiksi

k:n lähimmän naapurin päätössääntö on suboptimaalinen variaatio k:n lähimmän naapurin estimaatista:

- Etsi N:n opetusnäytteen joukosta luokiteltavan näytteen $\mathbf{x}$ k lähintä naapuria. Yleensä k on pariton eikä se ole luokkien lukumäärän M monikerta
- Laske kuinka moni (k_i) lähimmistä naapureista kuuluu luokkaan ω_i
- Valitse se luokka, jolle k_i on suurin

Lähimmän naapurin menetelmän luokitteluvirheelle P_{NN} voidaan laskea seuraavat teoreettiset rajat, kun $N \to \infty$:

$$P_{Bayes} \leq P_{NN} \leq P_{Bayes}(2 - \frac{M}{M-1}P_{Bayes}) \leq 2P_{Bayes}, \qquad (50)$$

missä P_{Bayes} on Bayes-säännön tuottama optimaalinen luokitteluvirhe ja M on luokkien lkm

Mikäli N on suuri, k:n lähimmän naapurin päätössääntö toimii paremmin isommilla k:n arvoilla kuin 1

Käytännössä k:n lähimmän naapurin sääntö toimii usein erittäin hyvin yksinkertaisuudestaan huolimatta, mutta on laskennallisesti raskas suurilla N:n arvoilla

6. LINEAARISET LUOKITTIMET

Edellisillä luennoilla tarkasteltiin luokitteluongelmaa tnjakaumien avulla ja esiteltiin menetelmiä, miten tarvittavat tnjakaumat voidaan estimoida. Tavoitteena oli löytää päätössääntö, joka minimoi luokkiteluvirhetn:n tai riskin

Tietyin tnjakaumista tehdyin oletuksin näin saatu luokitin on lineaarinen

Nyt ei oteta kantaa luokkiin liittyviin tnjakaumiin, vaan oletetaan, että kaikki havainnot voidaan luokitella oikein/riitävän hyvin lineaarisilla luokittimilla

Lineaaristen luokittimien etuna on niiden yksinkertaisuus ja laskennallinen keveys

6.1 Lineaariset diskriminanttifunktiot

Lineaarinen luokitin jakaa piirreavaruuden eri luokkia vastaaviksi päätösalueiksi hypertasojen avulla

Lineaarinen diskriminanttifunktio $g(\mathbf{x})$:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = 0, \tag{51}$$

missä $\mathbf{w} = [w_1, \ldots, w_l]$ on painovektori ja w_0 on kynnysarvo ('threshold')

Tarkastellaan kahta pistettä, $\mathbf{x}_1$ ja $\mathbf{x}_2$, diskriminattifunktion määrittelemällä hypertasolla:

$$\begin{aligned} 0 =& \mathbf{w}^T\mathbf{x}_1 + w_0 = \mathbf{w}^T\mathbf{x}_2 + w_0 \Rightarrow \\ & \mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = 0 \end{aligned} \tag{52}$$

Edellisestä nähdään, että painovektori $\mathbf{w}$ on ortogonaalinen hypertasoon nähden

Toisella puolella hypertasoa diskriminanttifunktio saa positiivisia arvoja, toisella puolella negatiivisia arvoja

Kahden luokan tapauksessa voidaan **luokittelusääntö** kirjoittaa seuraavasti:

- Merkitään $\tilde{\mathbf{x}} = [\mathbf{x}^T, 1]^T$ ja $\tilde{\mathbf{w}} = [\mathbf{w}^T, w_0]^T$
- Silloin $g(\mathbf{x}) \equiv g(\tilde{\mathbf{x}}) = \tilde{\mathbf{w}}^T\tilde{\mathbf{x}}$
- Valitaan luokka seuraavasti:

$$\begin{aligned} \omega_1 &: \ \tilde{\mathbf{w}}^T\tilde{\mathbf{x}} > 0 \\ \omega_2 &: \ \tilde{\mathbf{w}}^T\tilde{\mathbf{x}} < 0 \end{aligned} \tag{53}$$

Jos voidaan valita $\tilde{\mathbf{w}}$ siten, että edellinen päätössääntö ei tuota ainuttakaan luokitteluvirhettä, luokat ovat *lineaarisesti separoituvat*

Seuraavaksi käydään läpi menetelmiä, joiden avulla voidaan määrätä diskriminattifunktion painokertoimet

Tästä eteenpäin oletetaan, että piirrevektoreihin on liitetty loppuun vakiotermi, kuten edellä esitetyssä päätössäännössä, ellei toisin mainita

6.2 Perseptroni-algoritmi

(Perseptroni-algoritmin nimi tulee sitä, että se kehitettiin alunperin aivojen neuronimallien, 'perceptrons', opetukseen. Niistä lisää myöhemmin!)

Ol., että luokat ovat lineaarisesti separoituvia

Tarkastellaan kahden luokan tapausta

Valitaan diskriminattifunktion painokertoimet $\mathbf{w}$ ratkaisemalla seuraava optimointiongelma:

- Maksimoi (perseptroni)kustannusfunktio $J(\mathbf{w})$, joka on määritelty seuraavasti:
$$J(\mathbf{w}) = \sum_{\mathbf{x} \in Y} (\delta_x \mathbf{w}^T \mathbf{x}), \tag{54}$$
missä Y on niiden opetusnäytteiden osajoukko, jotka luokittuvat väärin $\mathbf{w}$ määrittämän hypertason perusteella. Muuttuja $\delta_x = -1$, kun $\mathbf{x} \in \omega_1$, ja $\delta_x = 1$, kun $\mathbf{x} \in \omega_2$
- Huom! $J(\mathbf{w})$ saa vain positiivisia arvoja. Sen arvo on nolla, jos ei synny

lainkaan luokitteluvirheitä

- Huom! $J(\mathbf{w})$ on jatkuva ja paloittain lineaarinen, $J(\mathbf{w})$:n gradientti ei ole määritelty niissä kohdin, joissa osajoukko Y muuttuu
- Edellisestä huolimatta, suoritetaan minimointi 'gradient descent'-henkisellä, iteratiivisellä menetelmällä:

$$\begin{aligned} \mathbf{w}(t+1) &= \mathbf{w}(t) - \rho_t \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}|_{\mathbf{w}(t)} \\ &= \mathbf{w}(t) - \rho_t \sum_{\mathbf{x} \in Y} \delta_x \mathbf{x} \end{aligned} \tag{55}$$

  missä on $\mathbf{w}(0)$ alustettu satunnaisesti ja ρ_t on positiivinen oppimiskerroin

Iterointia toistetaan, kunnes painovektorin arvo konvergoi ratkaisuun eli kaikki havainnot luokitellaan oikein

Ratkaisun löytyminen ja tarvittavien iteraatioaskelien lkm riippuu kertoimen ρ_t valinnasta

Vaikka $J(\mathbf{w})$:n gradientti ei ole määritelty kaikkialla, menetelmä löytää ratkaisun äärellisellä määrällä iteraatioaskelia, kun ρ_t valitaan seuraavasti:

$$\lim_{t\to\infty} \sum_{k=0}^{t} \rho_k = \infty$$

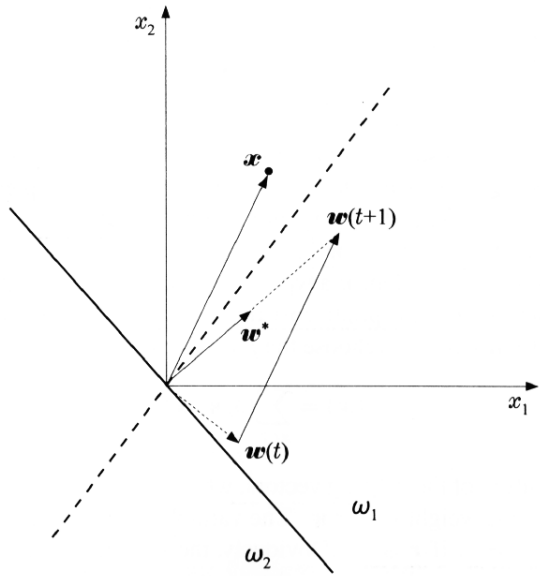
$$\lim_{t\to\infty} \sum_{k=0}^{t} \rho_k^2 < \infty$$

(todistus kirjassa)

Edelliset ehdot tarkoittavat käytännössä sitä, että ρ_t lähestyy nollaa, kun iteraatioaskelien lkm t kasvaa. Kerroin ρ_t ei saa kuitenkaan pienentyä liian nopeasti.

Sopivat arvot ρ_t:lle saadaan esim. seuraavasti: $\rho_t = c/t$, missä c on vakio. Kerroin ρ_t voi olla myös vakio, jos se on sopivasti rajoitettu

Perseptroni-säännön geometrinen tulkinta. Painovektorin päivitys, kun vain yksi piirrevektori luokittui väärin:



Perseptroni-säännön variaatioita

Edellä esitetyssä perseptroni-säännössä käytettiin jokaisella iteraatioaskeleella kaikkia N:ää opetusnäytettä. Painovektorin päivitys voidaan tehdä myös jokaisen havainnon perusteellä erikseen:

- Käydään opetusnäytteet läpi vuorotellen, päivitetään painovektoria seuraavasti:

$$\begin{aligned} &\mathbf{w}(t+1) = \mathbf{w}(t) + \rho\mathbf{x}_{(t)}, \quad \text{jos} \quad \mathbf{x}_{(t)} \in \omega_1 \quad \text{ja} \quad \mathbf{w}^T(t)\mathbf{x}_{(t)} \leq 0 \\ &\mathbf{w}(t+1) = \mathbf{w}(t) - \rho\mathbf{x}_{(t)}, \quad \text{jos} \quad \mathbf{x}_{(t)} \in \omega_2 \quad \text{ja} \quad \mathbf{w}^T(t)\mathbf{x}_{(t)} \geq 0 \\ &\mathbf{w}(t+1) = \mathbf{w}(t), \quad \text{muulloin} \end{aligned} \tag{56}$$

- Painovektoria päivitetään siis vain jos tarkasteltava opetusnäyte $\mathbf{x}_{(t)}$ luokittuu vääriin
- Opetusnäytteitä käydään läpi kunnes menetelmä konvergoi eli kaikki näytteet luokittuvat oikein
- Myös tämä menetelmä konvergoi äärellisellä määrällä iteraatioaskelia

Pocket-algoritmi

Ol., toisin kuin aikaisemmin, että luokat *eivät* ole lineaarisesti separoituvia

Seuraava menetelmä löytää lineaarisen diskriminanttifunktion, joka minimoi luokkiteluvirheiden lkm:n:

- Alusta $\mathbf{w}(0)$ satunnaisesti. Lisäksi, alusta 'varasto'-vektori $\mathbf{w}_s$ ja laskurimuuttuja h_s nolliksi
- Päivitä painovektoria käyttäen perseptroni-sääntöä (55)
- Laske kuinka monta (h) opetusnäytettä luokittuu oikein käytettäessä päivitettyä painovektoria $\mathbf{w}(t+1)$
- Jos $h > h_s$, aseta $\mathbf{w}_s = \mathbf{w}(t+1)$ ja $h_s = h$
- Toista kunnes konvergoi

Keslerin konstruktio

Edellä esitetyissä menetelmissä tarkasteltiin vain kahden luokan tapausta

Keslerin konstruktion avulla näitä menetelmiä voidaan käyttää myös $M > 2$:n luokan tapauksessa

Piirrevektorin $\mathbf{x}$ luokittelupäätös tehdään lineaaristen diskriminanttifunktioiden avulla seuraavasti:

$$\omega_i : \ \mathbf{w}_i^T \mathbf{x} > \mathbf{w}_j^T \mathbf{x}, \ \forall j \neq i \tag{57}$$

Muodostetaan luokan ω_i jokaista opetusnäytettä kohti seuraavat $(l+1)M \times 1$-ulotteiset vektorit: $\mathbf{x}_{ij} = [\mathbf{0}^T, \ldots, \mathbf{x}^T, \ldots, -\mathbf{x}^T, \ldots, \mathbf{0}^T]^T$, $i \neq j$. Vektorit koostuvat siis M:stä blokista, joista $i.$ ja $j.$ ovat $\mathbf{x}$ ja $-\mathbf{x}$ ja muut ovat nollavektoreita

Vastaavasti luokkakohtaiset painovektorit kootaan yhdeksi vektoriksi: $\mathbf{w} = [\mathbf{w}_1^T, \ldots, \mathbf{w}_M^T]^T$

Näiden avulla päätössääntö voidaan voidaan kirjoittaa uuteen muotoon:

$$\omega_i : \ \mathbf{w}^T \mathbf{x}_{ij} > 0 \ \forall j = 1, \ldots, M, \ j \neq i \tag{58}$$

Ongelmana on siis löytää painovektori $\mathbf{w}$, jonka positiivisella puolella ovat kaikki uudet vektorit $\mathbf{x}_{ij}$

Mikäli luokat ovat lineaarisesti separoituvia, voidaan painovektorin oppimiseen käyttää esim. perseptroni-sääntöä

Oppimisen jälkeen luokkakohtaiset painovektorit saadaan pilkkomalla $\mathbf{w}$ osiin

Huom! Vain painovektorin suunta on tärkeä, ei sen pituus. Edellä esitetyissä menetelmissä painovektorin pituus pyrkii kasvamaan: normalisoidaan painovektori yksikkövektoriksi jokaisen päivityksen jälkeen

6.3 Pienimmän neliön menetelmät

Usein tiedetään etukäteen, että luokat eivät ole lineaarisesti separoituvia, mutta silti halutaan käyttää luokitteluvirhetn:n kannalta suboptimaalista lineaarista luokitinta

Lineaarinen luokitin muodostetaan valitsemalla diskriminanttifunktion painovektori siten, että jokin ongelmaan sopiva kriteeri optimoituu

Yleensä määritellään diskriminanttifunktiolle tavoitearvot y erilaisille havainnoille $\mathbf{x}$ ja pyritään minimoimaan tavoitearvojen ja todellisten arvojen $g(\mathbf{x})$ neliöpoikkeamia ('Least Squares Methods')

Pienimmän neliösumman menetelmä

Pienimmän neliösumman menetelmässä minimoidaan kustannusfunktiota $J(\mathbf{w})$:

$$J(\mathbf{w}) = \sum_{i=1}^{N}(y_i - \mathbf{x}_i^T\mathbf{w})^2 \equiv \sum_{i=1}^{N} e_i^2, \tag{59}$$

missä N on opetusnäytteiden lkm ja y_i on $\mathbf{x}_i$:tä vastaava diskriminanttifunktion tavoitearvo, kahden luokan tapauksessa yleensä $y_i = \pm 1$

Kun derivoidaan $J(\mathbf{w})$ painovektorin $\mathbf{w}$ suhteen saadaan:

$$\begin{aligned} &\sum_{i=1}^{N}\mathbf{x}_i(y_i - \mathbf{x}_i^T\hat{\mathbf{w}}) = 0 \Rightarrow \\ &(\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T)\hat{\mathbf{w}} = \sum_{i=1}^{N}(\mathbf{x}_i y_i) \end{aligned} \tag{60}$$

Käytetään seuraavia merkintöjä:

$$\begin{aligned} \mathbf{X} &= [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T \\ \mathbf{y} &= [y_1, \ldots, y_N]^T \end{aligned} \tag{61}$$

Kaava (60) voidaan kirjoittaa silloin matriisimuodossa:

$$\begin{aligned} &(\mathbf{X}^T\mathbf{X})\hat{\mathbf{w}} = \mathbf{X}^T\mathbf{y} \Rightarrow \\ &\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}, \end{aligned} \tag{62}$$

missä matriisi $\mathbf{X}^T\mathbf{X}$ on otoskorrelaatiomatriisi ja $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ on $\mathbf{X}$:n pseudoinverssi

Esimerkki: pienimmän neliösumman luokitin Tarkastellaan kahden luokan tapausta, jossa piirrevektorit ovat kaksiulotteisia. Luokista on tehty seuraavat havainnot:

$$\begin{aligned}\omega_1 : &[0.2, 0.7]^T\ [0.3, 0.3]^T\ [0.4, 0.5]^T \\ &[0.6, 0.5]^T\ [0.1, 0.4]^T \\ \omega_2 : &[0.4, 0.6]^T\ [0.6, 0.2]^T\ [0.7, 0.4]^T \\ &[0.8, 0.6]^T\ [0.7, 0.5]^T\end{aligned}$$

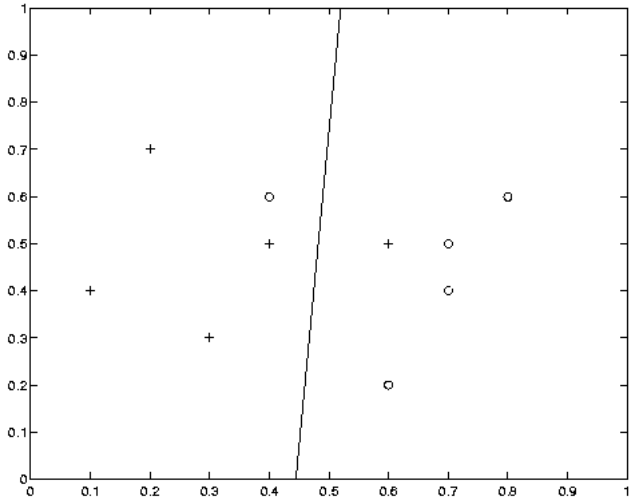
Luokat eivät ole lineaarisesti separoituvia. Yritetään löytää diskriminanttifunktio, joka on muotoa $g(\mathbf{x}; \mathbf{w}) = [\mathbf{x}, 1]^T\mathbf{w} = w_1x_1 + w_2x_2 + w_0$ ja joka minimoi virheiden neliösumman

Asetetaan diskriminattifunktion tavoitearvoksi 1 tai -1, kun havainto on luokasta ω_1 tai ω_2

Silloin

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 2.8 & 2.24 & 4.8 \\ 2.24 & 2.41 & 4.7 \\ 4.8 & 4.7 & 10 \end{bmatrix}$$
$$\mathbf{X}^T\mathbf{y} = [-1.6, 0.1, 0.0]^T$$

ja ratkaisuksi saadaan kaavan (62) perusteella $\hat{\mathbf{w}} = [-3.218, 0.241, 1.431]^T$


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

MSE-estimaatti

Seuraavaksi tarkastellaan neliöpoikkeamien summan sijasta neliöpoikkeaman odotusarvoa ('Mean Square Error Estimation')

Minimoitava kustannusfunktio $J(\mathbf{w})$:

$$J(\mathbf{w}) = \mathsf{E}[|y - \mathbf{x}^T\mathbf{w}|^2] \tag{63}$$

Kahden luokan tapauksessa, kun $y = \pm 1$, edellinen kaava voidaan kirjoittaa tnjakaumien avulla myös näin:

$$J(\mathbf{w}) = P(\omega_1)\int(1 - \mathbf{x}^T\mathbf{w})^2 p(\mathbf{x}|\omega_1)d\mathbf{x} + P(\omega_2)\int(1 + \mathbf{x}^T\mathbf{w})^2 p(\mathbf{x}|\omega_2)d\mathbf{x} \tag{64}$$

Välttämätön ehto $J(\mathbf{w})$:n minimille:

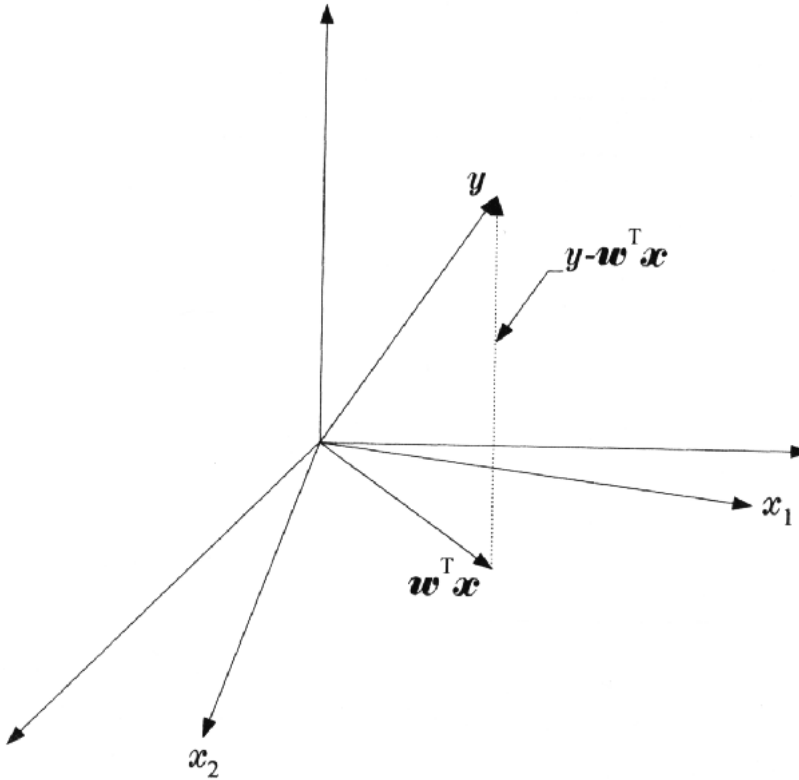
$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = 2\mathsf{E}[\mathbf{x}(y - \mathbf{x}^T\mathbf{w})] = \mathbf{0}, \tag{65}$$

Edellisestä kaavasta saadaan ratkaisu

$$\hat{\mathbf{w}} = \mathbf{R}_x^{-1}\mathrm{E}[\mathbf{x}y], \tag{66}$$

missä $\mathbf{R}_x$ on $\mathbf{x}$:n korrelaatiomatriisi ja $\mathrm{E}[\mathbf{x}y]$ on $\mathbf{x}$:n ja y:n ristikorrelaatiovektori

Ratkaisun geometrinen tulkinta? Diskriminanttifunktion tavoitearvoa approksimoidaan piirteiden lineaarikombinaatiolla, syntynyt virhe on ortogonaalinen piirreavaruuteen nähden:


y
y-w^T x
x_1
w^T x
x_2

Yleistys useammalle kuin kahdelle luokalle

Useamman ($M > 2$) kuin kahden luokan tapauksessa muodostetaan jokaiselle luokalle oma diskriminanttifunktio $g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x}$ ja asetetaan sen tavoitearvot y_i seuraavasti:

$$y_i = \begin{cases} 1, & \text{jos } \mathbf{x} \in \omega_i \\ 0, & \text{muulloin} \end{cases} \tag{67}$$

Huom! Kun $M = 2$, $y = \pm 1$ tuottaa saman ratkaisun kuin edellinen valinta, koska $\mathbf{w}^T\mathbf{x} = \frac{1}{2}(\mathbf{w}_1 - \mathbf{w}_2)^T\mathbf{x}$

Kootaan tavoitearvot vektoriksi $\mathbf{y} = [y_1, \ldots, y_M]^T$ ja painovektorit matriisiksi $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_M]$

Ratkaisu löytyy minimoimalla seuraava kustannusfunktio $J(\mathbf{W})$

$$J(\mathbf{W}) = \mathsf{E}[\|\mathbf{y} - \mathbf{W}^T\mathbf{x}\|^2] = \mathsf{E}[\sum_{i=1}^{M}(y_i - \mathbf{w}_i^T\mathbf{x})^2] \tag{68}$$

Edellisestä kaavasta nähdään, että voidaan suunnitella jokainen diskriminanttifunktio erikseen.

LMS- eli Widrow-Hoff algoritmi

Yleensä ei tunneta MSE-estimaatin ratkaisussa eli kaavassa (66) esiintyviä korrelaatiomatriisia $\mathbf{R}_x$ ja ristikorrelaatiovektoria $\mathrm{E}[\mathbf{x}y]$

Voidaan osoittaa, että ratkaisu ongelmaan, joka on muotoa $\mathrm{E}[F(\mathbf{x}_k, \mathbf{w})] = 0$, löytyy seuraavalla iteratiivisellä algoritmillä:

$$\hat{\mathbf{w}}(k) = \hat{\mathbf{w}}(k-1) + \rho_k F(\mathbf{x}_k, \hat{\mathbf{w}}(k-1)) \tag{69}$$

kunhan

$$\begin{aligned} \sum_{k=1}^{\infty} \rho_k &\to \infty \\ \sum_{k=1}^{\infty} \rho_k^2 &< \infty \end{aligned} \tag{70}$$

$\mathbf{x}_k$ on sarja satunnaisia vektoreita, jotka ovat peräisin samanlaisista tnjakaumista, $F(\cdot,\cdot)$ on jokin funktio, ja $\mathbf{w}$ on sen tuntematon parametrivektori

Kun edellistä sovelletaan diskriminattifunktion painokertoimien hakuun,

$$\hat{\mathbf{w}}(k) = \hat{\mathbf{w}}(k-1) + \rho_k \mathbf{x}_k (y_k - \mathbf{x}_k^T \hat{\mathbf{w}}(k-1)) \tag{71}$$

(kun $k \to \infty$, $\hat{\mathbf{w}}(k)$ lähestyy asymptoottisesti MSE-estimaattia)

Kerroin ρ_k voi olla myös sopivasti rajoitettu vakio $0 < \rho < 2/\text{trace}\{\mathbf{R}_x\}$. Voidaan osoittaa, että mitä pienempi ρ, sitä pienempi on estimaatin $\hat{\mathbf{w}}(k)$ varianssi. Toisaalta, konvergointi on hitaampaa pienellä ρ:lla

Kaavasta (71) nähdään, että estimaattia päivitetään jokaisen havainnon jälkeen. Kun ρ on vakio, pystyy estimaatti mukautumaan paremmin luokkien tnjakaumien muutoksiin

MSE-estimaatti ja luokkien a posteriori tn:t

Lähdetään tästä liikkeelle: voidaan helposti osoittaa, että

$$\begin{aligned}\hat{\mathbf{y}} &= \arg\min_{\tilde{\mathbf{y}}} \mathsf{E}[\|\mathbf{y} - \tilde{\mathbf{y}}\|^2] \\ &= \mathsf{E}[\mathbf{y}|\mathbf{x}] = \int_{-\infty}^{\infty} \mathbf{y} p(\mathbf{y}|\mathbf{x}) d\mathbf{y}\end{aligned} \tag{72}$$

(todistus kirjassa, odotusarvot lasketaan tnjakauman $p(\mathbf{y}|\mathbf{x})$ suhteen)

Edellinen tulos tarkoittaa sitä, että MSE-kriteerin mielessä paras mahdollinen estimaatti diskriminattifunktion tavoitearvolle $\mathbf{y}$ on sen odotusarvo annettuna $\mathbf{x}$.

Diskriminattifunktion voidaan ajatella olevan $\mathbf{y}$:n regressio annettuna $\mathbf{x}$

Muokataan kaavasta (68) yleisemmälle ongelmalle kustannusfunktio:

$$J = \mathsf{E}[\sum_{i=1}^{M} (g_i(\mathbf{x}; \mathbf{w}_i) - y_i)^2], \tag{73}$$

missä diskriminanttifunktiot eivät ole välttämättä lineaarisia $\mathbf{x}$:n tai parametriensä $\mathbf{w}_i$ suhteen

Edellinen kaava voidaan kirjoittaa myös näin:

$$J = \mathsf{E}[\sum_{i=1}^{M}(g_i(\mathbf{x};\mathbf{w}_i) - \mathsf{E}[y_i|\mathbf{x}])^2] + \mathsf{E}[\sum_{i=1}^{M}(\mathsf{E}[y_i^2|\mathbf{x}] - (\mathsf{E}[y_i|\mathbf{x}])^2)] \quad (74)$$

(välivaiheet kirjassa)

Jälkimmäinen termi ei riipu lainkaan diskriminanttifunktiosta, joten se voidaan jättää huomioimatta, kun minimoidaan J:tä

Nähdään kaavasta (72), että J minimoituu, kun $g_i(\mathbf{x};\hat{\mathbf{w}}_i)$ approksimoi MSE-mielessä mahdollisimman tarkasti $\mathsf{E}[y_i|\mathbf{x}]$:ää

Miksi tämä on tärkeä tulos?

Toisaalta:

$$\mathsf{E}[y_i|\mathbf{x}] = \sum_{j=1}^{M} y_i P(\omega_j|\mathbf{x}) \quad (75)$$

Kun valitaan tavoitearvot siten, että $y_i = 1$ tai $y_i = 0$ riippuen kuuluuko $\mathbf{x}$

luokkaan ω_i vai ei, $g_i(\mathbf{x}; \hat{\mathbf{w}}_i)$ on $P(\omega_i|\mathbf{x})$:n MSE-estimaatti

$P(\omega_i|\mathbf{x})$ voidaan siis estimoida valitsemalla diskriminanttifunktioiden parametrit MSE-kriteerin ja LMS-menetelmän avulla (tuntematta tnjakaumia!) ja sitä voidaan käyttää Bayesiläisessä luokittelussa

Se kuinka hyvin luokkien *a posteriori* tnjakaumien estimointi sitten onnistuu riippuu diskriminanttifunktioiden tyypistä

6.4 Fisherin diskriminantti

Eräs tapa muodostaa lineaarinen luokitin: etsitään suora, jolle projisoidut piirrevektorit separoituvat mahdollisimman hyvin, ja jaetaan suora sitten päätösalueiksi

Tarkastellaan kahden luokan tapausta

Määritellään projektio seuraavasti: $y = \mathbf{w}^T\mathbf{x}$, missä $\|\mathbf{w}\| = 1$

Sopiva mitta $J(\mathbf{w})$ luokkien separoituvuudelle?

Eräs järkevä mitta saadaan luokkien projektioden odotusarvojen $\mu_{Y_i} = \mathsf{E}[\mathbf{w}^T\mathbf{x}|\mathbf{x} \in \omega_i]$ ja varianssien $\sigma^2_{Y_i} = \mathsf{E}[\|\mathbf{w}^T\mathbf{x} - \mu_{Y_i}\|^2|\mathbf{x} \in \omega_i]$ avulla:

$$J(\mathbf{w}) = \frac{(\mu_{Y_1} - \mu_{Y_2})^2}{\sigma^2_{Y_1} + \sigma^2_{Y_2}} \tag{76}$$

tai otoskeskiarvojen $m_{Y_i} = 1/N_i \sum_{j=1}^{N_i} \mathbf{w}^T \mathbf{x}_j$ ja sironnan $s_{Y_i}^2 = \sum_{j=1}^{N_i} (\mathbf{w}^T \mathbf{x}_j - m_{Y_i})^2$ avulla:

$$J(\mathbf{w}) = \frac{(m_{Y_1} - m_{Y_2})^2}{s_{Y_1}^2 + s_{Y_2}^2} \tag{77}$$

(N_i on luokkaan ω_i kuuluvien opetusnäytteiden lkm)

Suoraa $y = \hat{\mathbf{w}}^T \mathbf{x}$, joka löytyy maksimoimalla $J(\mathbf{w})$, kutsutaan *Fisherin diskriminantiksi*

Tapaus 1: tunnetaan luokkien odotusarvot ja kovarianssimatriisit

Kun tunnetaan luokan ω_i odotusarvo μ_i ja kovarianssimatriisi $\mathbf{\Sigma}_i$, voidaan vastaavat tunnusluvut laskea projektioille: $\mu_{Y_i} = \mathbf{w}^T \mu_i$ ja $\sigma_{Y_i} = \mathbf{w}^T \mathbf{\Sigma}_i \mathbf{w}$

Derivoidaan $J(\mathbf{w})$ (76) $\mathbf{w}$:n suhteen ja asetetaan nollavektoriksi. Ratkaisuksi saadaan:

$$\hat{\mathbf{w}} = \frac{1}{2} k (\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2)^{-1} (\mu_1 - \mu_2), \tag{78}$$

missä

$$k = \frac{\sigma_{Y_1}^2 + \sigma_{Y_2}^2}{\mu_{Y_1} - \mu_{Y_2}} \tag{79}$$

normalisoi $\hat{\mathbf{w}}$:n pituuden 1:ksi

Tapaus 2: ei tunneta luokkien odotusarvoja ja kovarianssimatriiseja

Määritellään aluksi seuraavat matriisit:

$$\begin{aligned} \mathbf{S}_i &= \sum_{j=1}^{N_i} (\mathbf{x}_j - \mathbf{m}_i)(\mathbf{x}_j - \mathbf{m}_i)^T \\ \mathbf{S}_W &= \mathbf{S}_1 + \mathbf{S}_2 \\ \mathbf{S}_B &= (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T \end{aligned} \tag{80}$$

missä $\mathbf{m}_i = 1/N_i \sum_{j=1}^{N_i} \mathbf{x}_j$

Näiden avulla $J(\mathbf{w})$ (77) voidaan kirjoittaa seuraavasti:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \tag{81}$$

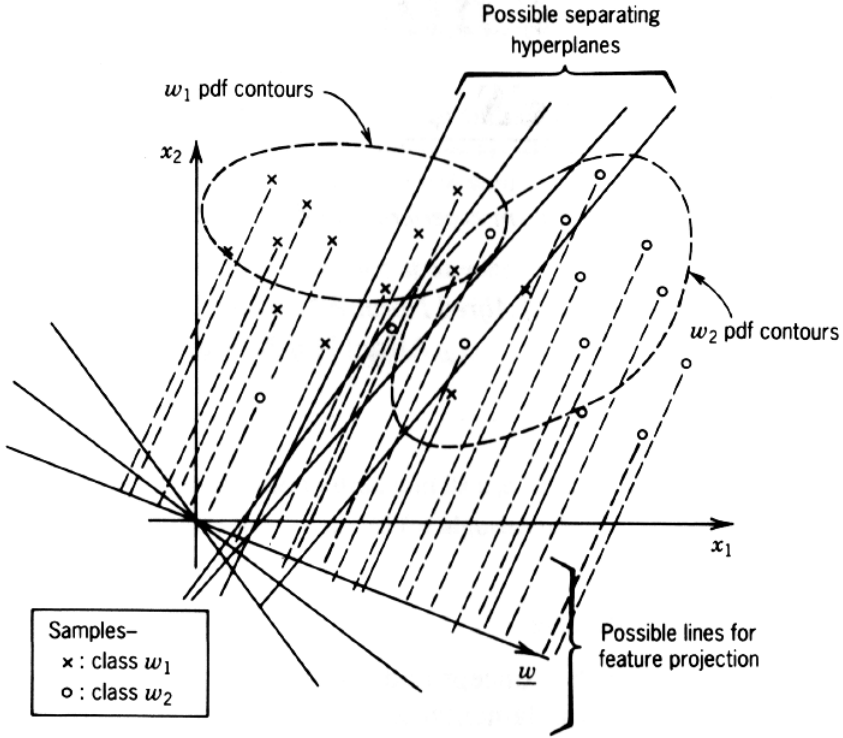
Derivoidaan $J(\mathbf{w})$ ja asetetaan nollavektoriksi. Tällöin saadaan seuraava yleinen ominaisvektoriongelma:

$$\lambda \mathbf{S}_W \hat{\mathbf{w}} = \mathbf{S}_B \hat{\mathbf{w}}, \tag{82}$$

ja ratkaisu:

$$\hat{\mathbf{w}} = \mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2) \tag{83}$$

(todistus kirjassa)


Possible separating
hyperplanes
w_1 pdf contours
x_2
w_2 pdf contours
x_1
Samples–
× : class w_1
o : class w_2
$\underline{w}$
Possible lines for
feature projection

7. EPÄLINEAARISET LUOKITTIMET

Yleensä luokitteluongelma ei ole lineaarisesti separoituva eivätkä lineaarisilla luokittimilla saadut ratkaisut ole tyydyttäviä

Tällä luennolla tarkastellaan menetelmiä, jotka jakavat piirreavaruuden *epälineaarisesti* eri luokkia vastaaviksi päätösalueksi

Aluksi annetaan teoreettinen motivointi epälineaarisille luokittimille, seuraavaksi käsitellään yleisimmin käytetyt menetelmät

Eräs perinteinen esimerkki epälineaarisesta luokitteluongelmasta on XOR-funktio

Esimerkki: XOR-ongelma

XOR ('Exclusive OR') on Boolean-funktio, jonka arvo on tosi (1) vain jos täsmälleen yksi sen argumenteistä on tosi. Muulloin sen arvo on epätosi (0)

Tarkastellaan 2-ulotteista tapausta. Silloin $\mathbf{x} = [x_1, x_2]^T,\ x_i \in \{0, 1\}$, ja luokkia ω_1 ja ω_2 vastaavat seuraavat havainnot:

$$\omega_1 :\ [1, 0]^T,\ [0, 1]^T \text{ (tosi)}$$
$$\omega_2 :\ [0, 0]^T,\ [1, 1]^T \text{ (epätosi)}$$

Kun piirretään luokitteluongelmasta kuva, nähdään helposti, että luokat eivät ole lineaarisesti separoituvia

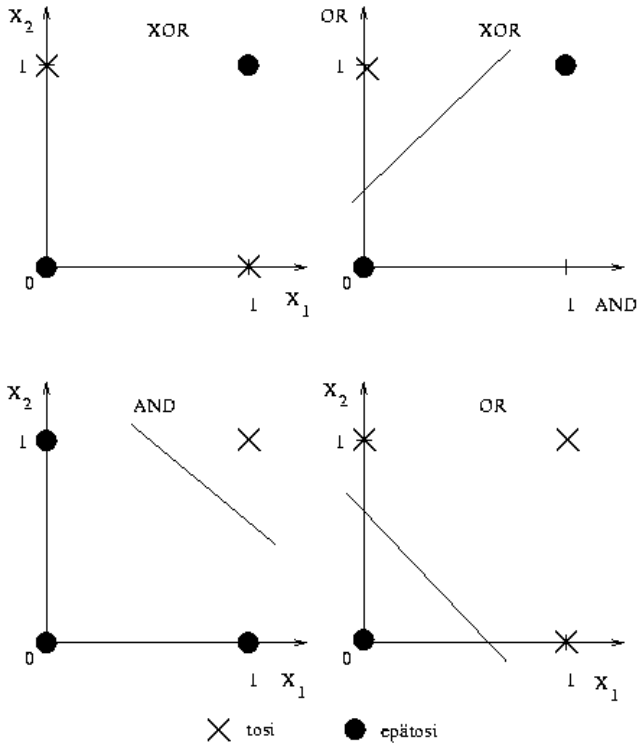
Toisaalta AND- ja OR-funktiota vastaavat luokitteluongelmat ovat lineaarisesti separoituvia

Jos tehdään piirteille epälineaarinen muunnos

$$[x_1, x_2]^T \rightarrow [\mathsf{AND}(x_1, x_2), \mathsf{OR}(x_1, x_2)]^T,$$

saadaan XOR-ongelmasta lineaarisesti separoituva

x_1	x_2	AND	class	OR	class	XOR	class
0	0	0	ω_2	0	ω_2	0	ω_2
0	1	0	ω_2	1	ω_1	1	ω_1
1	0	0	ω_2	1	ω_1	1	ω_1
1	1	1	ω_1	1	ω_1	0	ω_2


X2
XOR
1
0
1
X1
OR
XOR
1
0
1
AND
X2
AND
1
0
1
X1
X2
OR
1
0
1
X1
tosi
epätosi

7.1 Yleistetty lineaarinen luokitin

Tarkastellaan edellisessä esimerkissä esiteltyä tapaa tehdä luokitteluongelmasta lineaarisesti separoituva piirteiden epälineaarisen kuvauksen avulla yleisemmällä tasolla:

- Tarkastellaan kahden luokan tapausta: ω_1 ja ω_2
- Ol., että luokat ω_1 ja ω_2 eivät ole lineaarisesti separoituvia
- Ol., että piirrevektorit ovat l-ulotteisia ja niiden alkiot ovat reaalilukuja eli $\mathbf{x} \in \mathbb{R}^l$
- Suoritetaan piirrevektoreille seuraava epälineaarinen kuvaus:

$$\begin{aligned} &\mathbf{x} \in \mathbb{R}^l \rightarrow \mathbf{y} \in \mathbb{R}^k \\ &\mathbf{y} = [f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})]^T, \end{aligned} \tag{84}$$

  missä $f_i : \mathbb{R}^l \rightarrow \mathbb{R}$ on jokin epälineaarinen funktio

- Pyritään valitsemaan uusien piirteiden lkm k ja funktiot $f_i(\cdot)$ siten, että luokitteluongelmasta tulee lineaarisesti separoituva eli voidaan löytää seuraavanlainen separoiva hypertaso $\mathbf{w}$:

$$\begin{aligned} \omega_1 &: \ \mathbf{w}^T\mathbf{y} > 0 \\ \omega_2 &: \ \mathbf{w}^T\mathbf{y} < 0 \end{aligned} \tag{85}$$

Edellä kuvatun piirteiden epälineaaristen kuvausten seurauksena eri luokkiin liittyvät diskriminanttifunktiot ovat muotoa

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{k} w_i f_i(\mathbf{x}) = 0 \tag{86}$$

Uudessa piirreavaruudessa määriteltyjen hypertasojen painokertoimet voidaan etsiä käyttämällä edellisellä luennolla esiteltyjä menetelmiä

Kaava (86) voidaan tulkita diskriminanttifunktion approksimoinniksi interpolaatio- tai kantafunktioden $f_i(\cdot)$ avulla. Interpolaatiofunktiot voivat olla tyypiltään esim. eksponentiaalisia tai polynomeja

Coverin teoreema antaa teoreettisen perustelun, miksi tällainen piirteiden epälineaarinen kuvaus kannattaa tehdä

7.2 Coverin teoreema

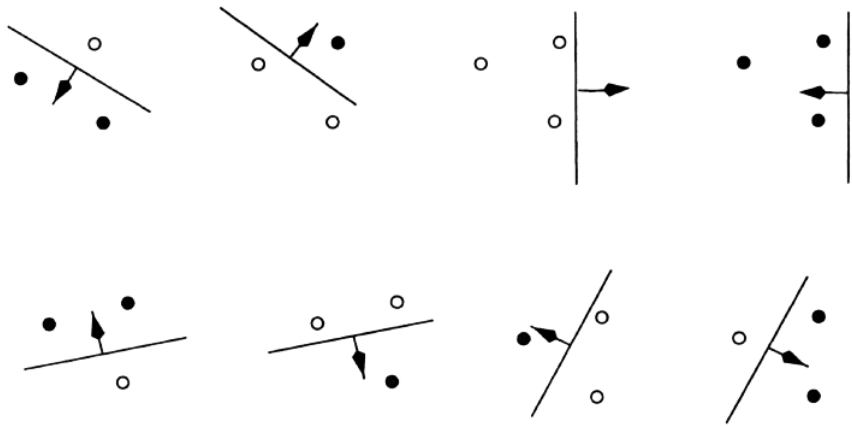
Todennäköisyys sille, että luokat ovat lineaarisesti separoituvia kasvaa, kun suoritetaan epälineaarinen kuvaus alkuperäisestä piirreavaruudesta uuteen, korkeampiulotteisempaan piirreavaruuteen (Cover, 1965)

Todistus:

- Voidaan osoittaa, että $(l-1)$-ulotteinen hypertaso voi jakaa N mielivaltaisesti valittua pistettä kahteen luokkaan $O(N,l)$:llä tavalla:

$$O(N,l) = 2\sum_{i=0}^{l} \binom{N-1}{i} \qquad \binom{N-1}{i} = \frac{(N-1)!}{(N-1-i)!i!} \tag{87}$$

Näitä erilaisia kahtiajakoja kutsutaan dikotomioiksi ('dichotomies'). Kun $N \leq l + 1$, $O(N, l) = 2^N$. Esim. $O(3, 2) = 8$

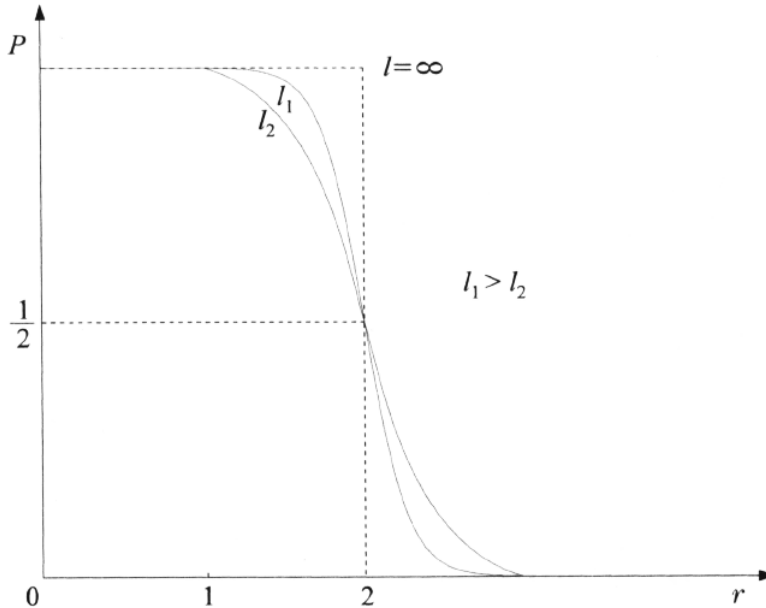


- Tn, että N:n pisteen dikotomia l-ulotteisessa piirreavaruudessa vastaa kahta lineaarisesti separoituvaa luokkaa on P_N^l:

$$P_N^l = \frac{O(N,l)}{2^N} = \begin{cases} \frac{1}{2^{N-1}} \sum_{i=0}^{l} \binom{N-1}{i}, & \text{kun } N > l+1 \\ 1, & \text{muulloin} \end{cases} \quad (88)$$

- Tarkastellaan P_N^l:n riippuvuutta N:stä ja l:stä muuttujan r avulla olettamalla, että $N = r(l+1)$. Kohdassa $r = 2$ $P_N^l = 1/2$. Lisäksi, kun $l \to \infty$, $P_N^l \to 1$, jos $N < 2(l+1)$

- Toisin sanoen, tn että luokat ovat lineaarisesti separoituvia lähestyy ykköstä, kun piirteiden lkm l lähestyy ääretöntä ja havaintojen lkm N on sopivasti rajoitettu


P
l=∞
l1
l2
l1 > l2
1/2
0
1
2
r

Edellisen tarkastelun perusteella piirrevektorit kannattaa kuvata korkeampiulotteiseen avaruuteen ja yrittää vasta sitten luokkien lineaarista separointia. Tn onnistumiselle on tällöin parempi

7.3 Polynomiluokitin

Tarkastellaan diskriminanttifunktioita, jotka on muodostettu kaavan (86) avulla ja käyttämällä kantafunktioina polynomeja:

- Ol., että polynomien asteluku r on 2. Silloin kaava (86) voidaan kirjoittaa seuraavasti:

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{l} w_i x_i + \sum_{i=1}^{l-1} \sum_{m=i+1}^{l} w_{im} x_i x_m + \sum_{i=1}^{l} w_{ii} x_i^2 = 0 \quad (89)$$

- Esim. jos $\mathbf{x} = [x_1, x_2]^T$, silloin $\mathbf{y} = [1, x_1, x_2, x_1x_2, x_1^2, x_2^2]^T$ ja

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{y} = 0, \tag{90}$$

  missä $\mathbf{w} = [w_0, w_1, w_2, w_{12}, w_{11}, w_{22}]^T$

- Vapaiden parametrien eli erilaisten muotoa $x_1^{p_1} \cdots x_l^{p_l}$, $0 \leq p_1 + \cdots + p_l \leq r$, olevien termien lkm k:

$$k = \frac{(l+r)!}{r!l!} \tag{91}$$

  Huom! Kun $l = 10$ ja $r = 10$, $k = 184\,756$

Vapaat parametrit voidaan taas valita esim. edellisellä luennolla esitetyillä menetelmillä

Esimerkki: XOR ja polynomiluokitin

Palataan takaisin XOR-ongelmaan ja muodostetaan sille polynomiluokitin

Valitaan kantafunktiot seuraavasti:

$$\mathbf{y} = [1, x_1, x_2, x_1 x_2]^T$$

Erilaisten havaintojen sijainnit uudessa piirreavaruudessa:

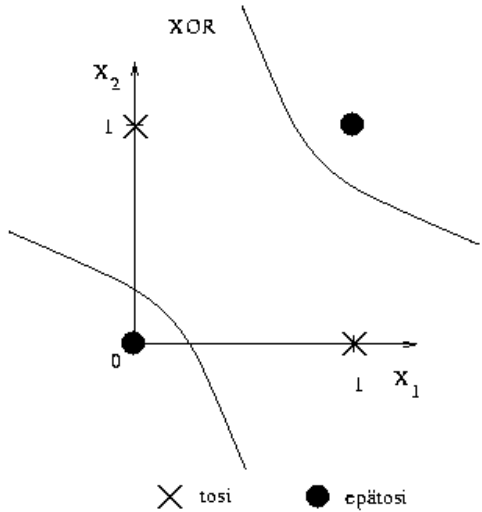
$$\omega_1 : \ [1,0]^T \rightarrow [1,1,0,0]^T, \ [0,1]^T \rightarrow [1,0,1,0]^T$$
$$\omega_2 : \ [0,0]^T \rightarrow [1,0,0,0]^T, \ [1,1]^T \rightarrow [1,1,1,1]^T$$

Nämä uudet pisteet separoituvat lineaarisesti esim. hypertason $\mathbf{w}^T\mathbf{y} = 0$ avulla, missä $\mathbf{w} = [-1/4, 1, 1, -2]^T$

Alkuperäisessä piirreavaruudessa tätä hypertasoa vastaa diskriminanttifunktio $g(\mathbf{x})$:

$$g(\mathbf{x}) = -\frac{1}{4} + x_1 + x_2 - 2x_1 x_2 = 0, \tag{92}$$

$g(\mathbf{x})$ näyttää suurinpiirtein tältä:



7.4 RBF-luokitin

(RBF-luokitin voidaan ymmärtää myös neuroverkkona...)

Seuraavaksi muodostetaan diskriminanttifunktiot käyttäen kaavaa (86) ja kantafunktiota, jotka ovat muotoa $f_i(\|\mathbf{x} - \mathbf{c}_i\|)$

Kantafunktioiden arvo siis riippuu vain piirrevektorin $\mathbf{x}$ ja keskusvektorin $\mathbf{c}_i$ välisestä Euklidisesta etäisyydestä. Siitä kantafunktioiden englanninkielinen nimitys, 'Radial Basis Functions' (RBF)

Kantafunktiot voivat olla tyypiltään esim. seuraavanlaisia:

$$f_i(\mathbf{x}) = \exp(-\frac{1}{2\sigma_i^2}\|\mathbf{x} - \mathbf{c}_i\|^2) \tag{93}$$

$$f_i(\mathbf{x}) = \frac{\sigma_i^2}{\sigma_i^2 + \|\mathbf{x} - \mathbf{c}_i\|}, \tag{94}$$

joista ensimmäinen, gaussinen, on yleisemmin käytetty

Kun käytetään gaussisia kantafunktiota, kaava (86) voidaan kirjoittaa näin:

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{k} w_i \exp(-\frac{(\mathbf{x}-\mathbf{c}_i)^T(\mathbf{x}-\mathbf{c}_i)}{2\sigma_i^2}) = 0 \qquad (95)$$

Edellisen kaavan voi tulkita siten, että diskriminanttifunktiota approksimoidaan summaamalla yhteen keskusvektoreiden ympärille asetettuja kumpareita. Kumpareiden leveyttä voidaan säätää muuttamalla σ_i:ja

RBF-luokittimella ja tnjakauman estimoinnilla Partzen-ikkunoilla on paljon yhteistä. Jälkimmäisessä menetelmässä jokaisen opetusnäytteen päälle asetetaan kumpare, ensimmäisessä vain keskusvektoreiden (k kpl:tta) päälle

Kuinka valitaan sopivat parametrit?

- RBF-luokittimille pitää valita seuraavat parametrit: k, $\mathbf{w} = [w_0, \ldots, w_k]^T$, $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_k]$ ja $\sigma = [\sigma_1, \ldots, \sigma_k]^T$
- Usein valitaan ensin $\mathbf{C}$ ja σ, ja sitten painokertoimet $\mathbf{w}$ käyttäen esim. edellisellä luennolla esiteltyjä menetelmiä

- k:n, $\mathbf{C}$:n ja σ:n valinta voi perustua esim. ongelmaan liittyvään *a priori* tietoon tai klusterointimenetelmillä (näistä myöhemmin) saatuihin tuloksiin

- Mikäli opetusjoukko on edustava otos kaikista mahdollisista piirrevektoreista ja hyvin jakautunut, voidaan keskusfunktiot myös arpoa niiden joukosta

- Parametrit (paitsi k) voidaan myös valita minimoimalla seuraavan tyyppistä kustannusfunktiota $J(\mathbf{w}, \mathbf{C}, \sigma)$:

$$J(\mathbf{w}, \mathbf{C}, \sigma) = \sum_{j=1}^{N} \Phi(y(j) - g(\mathbf{x}(j))), \tag{96}$$

  missä $y(j)$ on opetusnäytettä $\mathbf{x}(j)$ vastaava diskriminanttifunktion tavoitearvo ja $\Phi(\cdot)$ jokin differentioituva funktio, esim. $(\cdot)^2$

- Edellä esitetty ongelma on tyypillinen epälineaarinen optimointiongelma, jonka ratkaisu löytyy parametrien suhteen lasketun J:n gradientin

nollakohdasta. Käytännössä tällaisen ongelman ratkaisu ei ole täysin mutkatonta

Esimerkki: XOR ja RBF-luokitin

Lähestytään jo tutuksi käynyttä XOR-ongelmaa käyttäen RBF-luokitinta

Tehdään seuraavanlainen epälineaarinen kuvaus piirrevektoreille:

$$\mathbf{y} = \mathbf{y}(\mathbf{x}) = \begin{bmatrix} \exp(-\|\mathbf{x} - \mathbf{c}_1\|^2) \\ \exp(-\|\mathbf{x} - \mathbf{c}_2\|^2) \end{bmatrix},$$

missä $\mathbf{c}_1 = [1, 1]^T$ ja $\mathbf{c}_2 = [0, 0]^T$

Tällöin piirrevektorit kuvautuvat seuraaviin paikkoihin uudessa piirreavaruudessa:

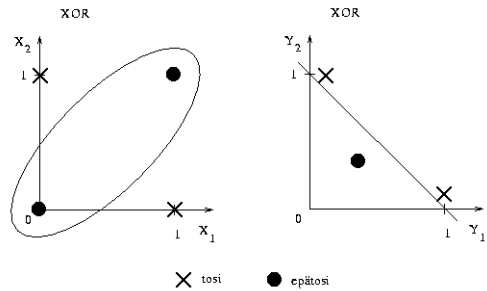
$$\omega_1 : \ [1, 0]^T \rightarrow [0.368, 0.368]^T, \ [0, 1]^T \rightarrow [0.368, 0.368]^T$$
$$\omega_2 : \ [0, 0]^T \rightarrow [0.135, 1]^T, \ [1, 1]^T \rightarrow [1, 0.135]^T$$

ja luokista tule lineaarisesti separoituvat

Valitaan separoiva hypertaso uudessa piirreavaruudessa seuraavasti:

$$g(\mathbf{y}) = y_1 + y_2 - 1 = \\ \exp(-\|\mathbf{x} - \mathbf{c}_1\|^2) + \exp((-\|\mathbf{x} - \mathbf{c}_2\|^2) - 1 = 0$$

Luokitteluongelman ratkaisu uudessa ja alkuperäisessä piirreavaruudessa:



Alkuperäisessä piirreavaruudessa päätösraja ei ole täsmälleen ellipsi kuten kuvassa, vaan vain sitä muistuttava käyrä. Kirjassa on parempi kuva

7.5 Tukivektorikone

('A Tutorial on Support Vector Machines for Pattern Recognition', http://svm.research.bell-labs.com)

Tukivektorikoneen ('Suport Vector Machines', SVM) idea on hyvin samankaltainen kuin RBF-luokittimen

SVM-menetelmä perustuu Vapnik-Chernovenkis oppimisteoriaan ja rakenteellisen riskin minimointiin (Nämä asiat opetetaan esim. neuraalilaskennan jatkokurssilla)

SVM-menetelmässä piirrevektoreille suoritetaan aluksi epälineaarinen kuvaus ja pyritään sitten löytämään yhdensuuntaiset hypertasot, jotka maksimoivat luokkien väliin jäävän marginaalin ja minimoivat luokitteluvirheet

Tarkastellaan aluksi lineaarista SVM-menetelmää lineaarisesti separoituvalle ongelmalle, tehdään sitten menetelmälle epälineaarinen yleistys ja tarkastellaan lopuksi ongelmaa, joka ei ole lineaarisesti separoituva

Lineaarisesti separoituvat luokat

Tarkastellaan kahden luokan tapausta

Ol., että luokat ovat lineaarisesti separoituvat

Etsitään kaksi yhdensuuntaista hypertasoa, H_1 ja H_2, siten, että seuraavat ehdot toteutuvat kaikille opetusnäytteille $\mathbf{x}(i)$, $0 \leq i \leq N$:

$$\begin{aligned} \mathbf{w}^T\mathbf{x}(i) + w_0 &\geq +1, \quad \text{kun} \quad \mathbf{x}(i) \in \omega_1 \\ \mathbf{w}^T\mathbf{x}(i) + w_0 &\leq -1, \quad \text{kun} \quad \mathbf{x}(i) \in \omega_2, \end{aligned} \tag{97}$$

missä $\mathbf{w}$ on hypertasojen normaali

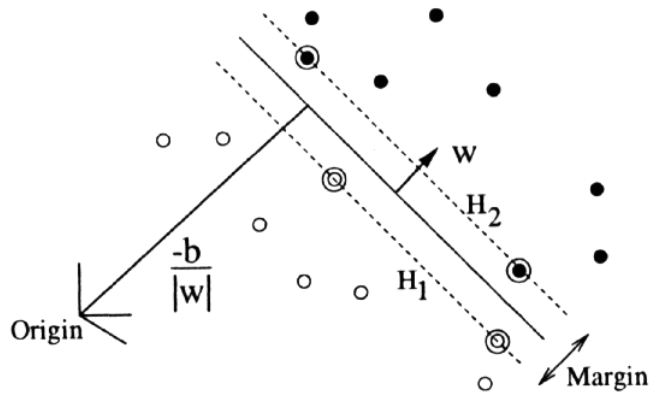
Hypertasot on määritelty seuraavasti:

$$\begin{aligned} H_1 &: \ \mathbf{w}^T\mathbf{x} + w_0 = +1 \\ H_2 &: \ \mathbf{w}^T\mathbf{x} + w_0 = -1 \end{aligned} \tag{98}$$

Ehdot (97) voidaan yhdistää muuttujien $y(i)$ avulla, jotka saavat arvon $+1$ tai -1 riippuen siitä kuuluuko $\mathbf{x}(i)$ luokkaan ω_1 vai ω_2

$$y(i)(\mathbf{w}^T\mathbf{x}(i) + w_0) - 1 \geq 0, \forall 1 \leq i \leq N \tag{99}$$

Koska luokat oletettiin lineaarisesti separoituviksi, hypertasojen H_1 ja H_2 väliin jää marginaali, jossa ei ole lainkaan opetusnäytteitä



Olkoot d_+ (d_-) etäisyys hypertason $\mathbf{w}^T\mathbf{x} + w_0 = 0$ ja sitä lähinnä olevan luokan ω_1 (ω_2) pisteen välinen Euklidinen etäisyys. Silloin marginaalin leveys on $d_+ + d_- = 2/\|\mathbf{w}\|$

Yritetään etsiä sellaiset hypertasot H_1 ja H_2, että marginaali maksimoituu (tai $\|\mathbf{w}\|^2$ minimoituu) ja ehdot (99) toteutuvat

Muodostetaan minimoitava kustannusfunktio $J(\mathbf{w}, w_0, \alpha)$ käyttäen Lagrangen kertoimia $\alpha = [\alpha_1, \ldots \alpha_N]^T \geq \mathbf{0}$:

$$J(\mathbf{w}, w_0, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i(y(i)(\mathbf{w}^T\mathbf{x}(i) + w_0) - 1) \qquad (100)$$

Minimiratkaisu löytyy J:n gradientin nollakohdasta:

$$\frac{\partial J}{\partial \mathbf{w}} = \mathbf{0} \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i y(i) \mathbf{x}(i) \tag{101}$$

$$\frac{\partial J}{\partial w_0} = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y(i) = 0 \tag{102}$$

Sijoitetaan nämä ehdot kaavaan (100) ja muodostetaan ns duaaliongelma α:n löytämiseksi:

$$\max_{\alpha} Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y(i) y(j) \mathbf{x}^T(i) \mathbf{x}(j) \tag{103}$$

siten, että

$$\sum_{i=1}^{N} \alpha_i y(i) = 0 \tag{104}$$

$$\alpha_i \geq 0, \forall 0 \leq i \leq N \tag{105}$$

Kun on löydetty duaaliongelman ratkaisu α^*, saadaan optimaaliset $\mathbf{w}^*$ ja w_0^* seuraavasti:

$$\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i^* y(i) \mathbf{x}(i) \tag{106}$$

$$w_0^* = 1 - \mathbf{x}^T(i)\mathbf{w}^*, \quad \text{missä} \quad \mathbf{x}(i) \in \omega_1, \ \alpha_i^* > 0 \tag{107}$$

Saadaan siis ratkaisuksi marginaalin maksimoiva diskriminanttifunktio $g(\mathbf{x})$, joka kulkee hypertasojen H_1 ja H_2 puolessavälissä:

$$g(\mathbf{x}) = \mathbf{x}^T\mathbf{w}^* + w_0^* = \sum_{i=1}^{N} \alpha_i^* y(i)\mathbf{x}^T(i)\mathbf{x} + w_0^* = 0 \qquad (108)$$

Edellinen kaava voidaan tulkita siten, että diskriminanttifunktiota approksimoidaan lineaarisilla kantafunktiolla $\mathbf{x}^T(i)\mathbf{x}$

Niitä opetusnäytteitä $\mathbf{x}(i)$, joita vastaava Lagrangen kerroin $\alpha_i > 0$, kutsutaan *tukivektoreiksi*. Ainoastaan tukivektorit huomioidaan diskriminanttifunktiossa (vrt RBF: automaattinen kantafunktioden valinta!)

Epälineaarinen yleistys

SVM-menetelmälle saadaan epälineaarinen yleistys korvaamalla kaavassa (108) esiintyvät sisätulot $\mathbf{x}^T(i)\mathbf{x}$ epälineaarisilla kantafunktioilla

$$K(\mathbf{x}, \mathbf{x}(i)) = \mathbf{\Phi}(\mathbf{x})^T \mathbf{\Phi}(\mathbf{x}(i)), \tag{109}$$

missä $\mathbf{\Phi}(\cdot) : \mathbb{R}^l \rightarrow \mathbb{R}^k$ on jokin epälineaarinen kuvaus

Tällainen ratkaisu saadaan, jos ensiksi kuvataan piirrevektorit funktion $\mathbf{\Phi}(\cdot)$ avulla uuteen piirreavaruuteen ja muodostetaan sitten lineaarinen SVM

Epälineaarisen SVM:n ratkaisussa esiintyviä sisätuloja ei välttämättä tarvitse laskea uudessa piirreavaruudessa

Kantafunktioiden tulee toteuttaa Mercerin teoreeman ehdot. Mm seuraavat funktiot ovat käypiä kantafunktioita:

$$(\mathbf{x}^T\mathbf{x}(i) + 1)^p \quad \text{(polynomiaalinen SVM)} \tag{110}$$

$$\exp(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{x}(i)\|^2) \quad \text{(RBF SVM)} \tag{111}$$

Lineaarisesti separoitumattomat luokat

Tarkastellaan edelleen kahden luokan tapausta, mutta oletetaan, että luokat eivät ole lineaarisesti separoituvat

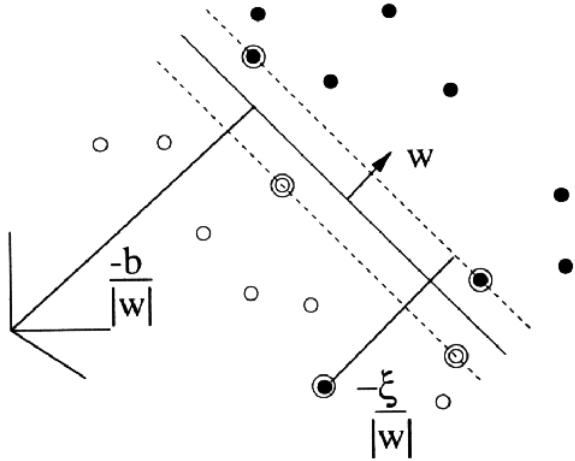
Nyt pyritään maksimoimaan luokkien väliin jäävää marginaalia ja minimoimaan luokitteluvirheiden lkm:ää

Muodostetaan opetusnäytteille seuraavat ehdot:

$$y(i)(\mathbf{w}^T\mathbf{x}(i) + w_0) \geq 1 - \xi_i \tag{112}$$

missä $\xi_i \geq 0$ on ns slack-muuttuja, joka kertoo onko opetusnäyte $\mathbf{x}(i)$ marginaalin sisällä ($0 \leq \xi_i \leq 1$) tai väärällä puolella ($\xi_i > 1$)

$\sum_{i=1}^{N} \xi_i$ on yläraja luokitteluvirheiden lkm:lle


w
-b
|w|
-ξ
|w|

Minimoitava kustannusfunktio, jossa edelliset ehdot on huomioitu käyttäen Lagrangen kertoimia $\alpha \geq \mathbf{0}$ ja $\mu \geq \mathbf{0}$:

$$\begin{aligned} J(\mathbf{w}, w_0, \xi, \alpha, \mu) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \\ &- \sum_{i=1}^{N}\alpha_i(y(i)(\mathbf{w}^T\mathbf{x}(i) + w_0) - 1 + \xi_i) - \sum_{i=1}^{N}\mu_i\xi_i \end{aligned} \tag{113}$$

missä $\xi = [\xi_1, \ldots, \xi_N]^T$, $\mu = [\mu_1, \ldots, \mu_N]^T$ ja C ovat käyttäjän määräämiä parametrejä

Kun toimitaan kuten lineaarisesti separoituvassa tapauksessa, saadaan seuraava duaaliongelma:

$$\max_{\alpha} Q(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y(i)y(j)\mathbf{x}^T(i)\mathbf{x}(j) \tag{114}$$

siten, että

$$\sum_{i=1}^{N} \alpha_i y(i) = 0 \tag{115}$$

$$0 \leq \alpha_i \leq C,\ \forall 0 \leq i \leq N \tag{116}$$

Kun on löydetty duaaliongelman ratkaisu α^*, optimaalinen $\mathbf{w}^*$ saadaan seuraavasti:

$$\mathbf{w}^* = \sum_{i=1}^{N_s} \alpha_i^* y(i) \mathbf{x}(i), \tag{117}$$

missä N_s on tukivektoreiden lkm

Loput tuntemattomat eli w_0^*, ξ^* ja μ^* voidaan ratkaista seuraavista yhtälöistä:

$$\alpha_i^*(y(i)(\mathbf{x}^T(i)\mathbf{w}^* + w_0) - 1 + \xi_i) = 0, \forall 0 \leq i \leq N \tag{118}$$

$$\mu_i \xi_i = 0, \forall 0 \leq i \leq N \tag{119}$$

$$\alpha_i^* + \mu_i = C, \forall 0 \leq i \leq N \tag{120}$$

Tukivektoreita ovat ne opetusnäytteet $\mathbf{x}(i)$, joille pätee:

$$y(i)(\mathbf{x}^T(i)\mathbf{w}^* + w_0^*) = 1 - \xi_i^* \tag{121}$$

eli $0 < \alpha_i \leq C$

Epälineaarinen yleistys muodostetaan samalla tavalla kuin lineaarisesti separoituvassa tapauksessa korvaamalla ratkaisussa esiintyvät sisätulot $\mathbf{x}^T(i)\mathbf{x}$ epälineaarisilla kantafunktioilla

Mikäli luokkia on enemmän kuin kaksi, muodostetaan SVM jokaisen luokkaparin välille

SVM-menetelmän etuna on se, että joudutaan ratkaisemaan vain kvadraattinen optimointiongelma, jossa rajoitusehdot ovat lineaarisia

SVM-menetelmässä kantafunktioiden lkm:n ja sijoittelun valinta tapahtuu automaattisesti

Toisaalta, jos luokat ovat kovin päällekkäiset tai datassa on paljon ns outlier-pisteitä, menetelmä löytää runsaasti tukivektoreita

Esimerkki: XOR ja SVM-menetelmä

Ratkaistaan jälleen kerran XOR-ongelma, mutta nyt epälineaarisella SVM-menetelmällä

Merkitään +1=tosi, -1=epätosi eli:

i	$\mathbf{x}(i)$	class	$y(i)$
1	$[-1,-1]^T$	ω_2	-1
2	$[-1,+1]^T$	ω_1	+1
3	$[+1,-1]^T$	ω_1	+1
4	$[+1,+1]^T$	ω_2	-1

Käytetään polynomiaalisia kantafunktioita, joiden asteluku on 2:

$$\begin{aligned} K(\mathbf{x},\mathbf{x}(i)) =& (\mathbf{x}^T\mathbf{x}(i)+1)^2 \\ =& 1 + x_1^2 x_1^2(i) + 2x_1x_2x_1(i)x_2(i) \\ & + x_2^2 x_2^2(i) + 2x_1x_1(i) + 2x_2x_2(i) \end{aligned}$$

Tämä kantafunktio vastaa seuraavaa piirteiden epälinearista kuvausta $\mathbf{f}: \mathbf{x} \in \mathbb{R}^2 \rightarrow \mathbf{y} \in \mathbb{R}^6$

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) = [1, x_1^2, \sqrt{2}x_1x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2]^T$$

Koska uusissa piirrevektoreissa oli mukana vakiotermi 1, kynnysarvo w_0 on turha ja se voidaan asettaa suoraan nollaksi

Tehdään ratkaisulle (114) epälineaarinen yleistys korvaamalla alkuperäiset piirrevektorit uusilla piirrevektoreilla

Uusien piirrevektoreiden sisätulot voidaan korvata kantafunktioilla

Silloin saadaan seuraava duaaliongelman kustannusfunktio:

$$\begin{aligned} Q(\alpha) =& \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}(9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 \\ &+ 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2) \end{aligned}$$

$Q(\alpha)$:n maksimi löytyy gradientin nollakohdasta:

$$\frac{\partial Q(\alpha)}{\partial \alpha} = \mathbf{0} \Rightarrow \alpha^* = \frac{1}{8}[1,1,1,1]^T$$

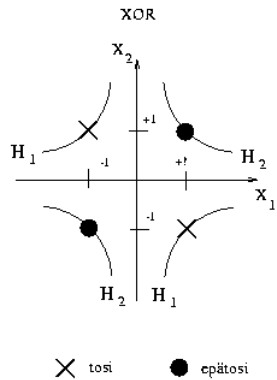
Edellisen tuloksen perusteella tiedetään, että kaikki opetusnäytteet ovat tukivektoreita. Silloin kaavan (117) epälineaarisen yleistyksen perusteella

$$\begin{aligned}\mathbf{w}^* =& \sum_{i=1}^{N_s} \alpha_i^* y(i) \mathbf{f}(\mathbf{x}(i)) \\ =& [0,0,-1/\sqrt{2},0,0,0]^T\end{aligned}$$

Optimaalinen diskriminanttifunktio on siis

$$g(\mathbf{x}; \mathbf{w}^*) = \mathbf{f}^T(\mathbf{x})\mathbf{w}^* = 0 \Rightarrow -x_1 x_2 = 0$$

ja sen antama ratkaisu näyttää alkuperäisessä piirreavaruudessa tältä:



8. NEUROVERKKOMENETELMÄT

Ihmisten ja eläinten loistava hahmontunnistuskyky perustuu lukuisiin yksinkertaisiin aivosoluihin ja niiden välisiin kytkentöihin

Mm edellisen innoittamana on kehitelty laskennallisia menetelmiä, joita kutsutaan seuraavilla nimillä: '(Artificial) Neural Networks' eli (A)NN, 'Connectionist Modelling', 'Neuromorphic Modelling' ja 'Parallel Distributed Processing' eli PDP

Neuroverkkomenetelmien tärkeimmät ominaispiirteet ovat:

- Yksinkertaiset laskentayksiköt, neuronit tai perseptronit
- Verkkomainen struktuuri, synaptiset kytkennät
- Rinnakkainen laskenta
- Black Box-malli syötteen ja vasteen välille

Neuroverkko koostuu toisiinsa (lokaalisti) kytketyistä (lukuisista) neuroneista

Kytkentöjen vahvuus on säädeltävissä synaptisten painojen avulla

Yksittäisen neuronin laskentakyky on vähäinen

Neuronien väliset kytkennät (neuroverkon struktuuri) voivat jakaa monimutkaisen ongelman yksinkertaisiksi osaongelmiksi

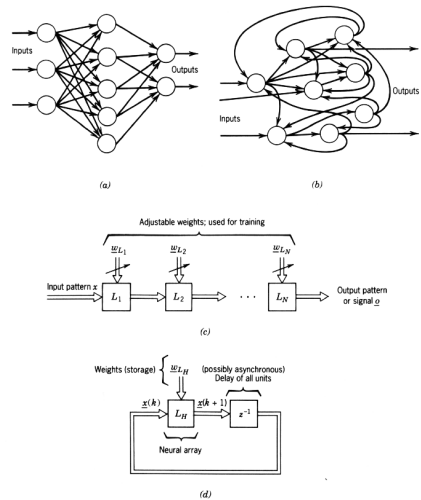
Kokoinaslaskenta-aika nopeutuu, kun osaongelmat ratkaistaan rinnakkaisesti

Neuroverkot ovat dynaamisia systeemejä, joiden tila (neuronien kytkennät ja ulostulo) muuttuu ajan, sisääntulojen ja alkutilan mukaan

Hahmontunnistusongelmassa pyritään löytämään neuronien välille sellaiset kytkennät, että syntyy haluttu assosiatiivinen käyttäytyminen: annettu sisääntulo (havainnot) yhdistetään tiettyyn vasteeseen (luokitus)

Klusterointiongelmassa pyritään muodostamaan neuroverkon avulla malli havaintojen rakenteelle (esim. SOM)

Erilaisia neuroverkkojen rakenteita: (a) Feedforward MLP-verkko, (b) rekurentti Hopfield-verkko, (c) Yleinen kaaviokuva (a):sta, (d) yleinen kaaviokuva (b):stä



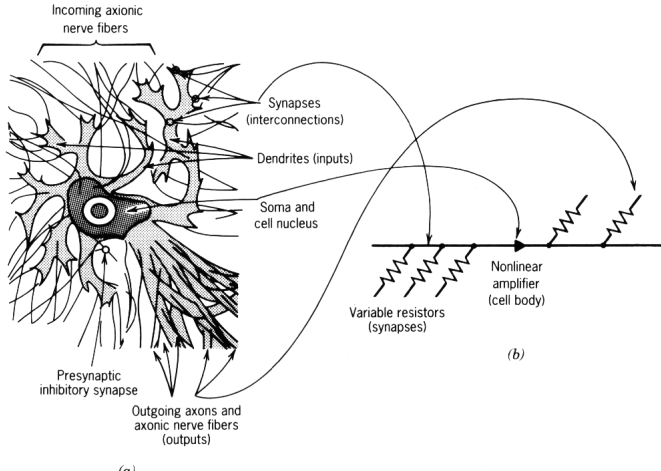
Tyypillisiä piirteitä neuroverkkosovellutuksille:

- Korkeaulotteinen piirreavaruus
- Monimutkaiset riippuvuussuhteet piirteiden välillä
- Tyhjä, yksikäsitteinen tai (yleensä) usean tasavertaisen vaihtoehdon muodostama ratkaisujoukko

Hahmontunnistuksessa neuroverkot soveltuvat varsinaisen luokittelun ja vertailun lisäksi esikäsittelyyn ja piirreirrotukseen

8.1 Perseptroni

Biologinen (a) ja elektroninen (b) neuronimalli:

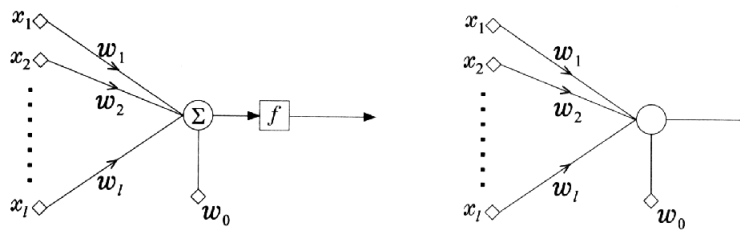


Perseptroni on malli neuroverkon neuronille, joka suorittaa sisääntulolleen $\mathbf{x}$ seuraavan, yksinkertaisen laskutoimituksen:

$$v = w_0 + \sum_{i=1}^{l} w_i x_i = w_0 + \mathbf{w}^T \mathbf{x}, \tag{122}$$

$$y = f(v), \tag{123}$$

missä y on neuronin ulostulo, $\mathbf{w}$ synaptisten kytkentöjen painokertoimet, w_0 kynnysarvo ja $f(\cdot)$ aktivaatiofunktio



Perseptronin aktivaatiofunktio on usein tyypiltään jokin seuraavista:

$$f(x) = \begin{cases} -1, & \text{jos } x < 0 \\ +1, & \text{jos } x > 0 \end{cases}, \tag{124}$$

$$f(x) = \begin{cases} 0, & \text{jos } x < 0 \\ 1, & \text{jos } x > 0 \end{cases}, \tag{125}$$

$$f(x) = \frac{1}{1 + \exp(-ax)}, \tag{126}$$

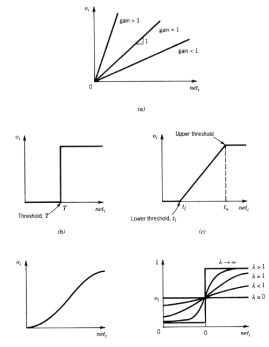
$$f(x) = c\frac{1 - \exp(-ax)}{1 + \exp(-ax)} = c\tanh(\frac{ax}{2}), \tag{127}$$

missä a ja c ovat funktion muotoa säätelevi parametrejä

Edelliset aktivaatiofunktiot ovat tyypiltään epäjatkuvia askelfunktioita (McCulloch-Pitts perseptroni) tai jatkuvia ja derivoituvia litistysfunktioita ('squashing function')

Molemmilla tyypeillä on puolensa: askelfunktioilla neuroverkon suorittaman kuvauksen analysointi, ja jatkuvilla funktiolla neuroverkon opettaminen, on helpompaa

Erilaisia aktivaatiofunktiota: (a) lineaarinen, (b) kynnys/askelfunktio, (c) lineaarinen kynnysfunktio, (d) sigmoidi, (f) sigmoideja erilaisilla parametrin arvoilla



Kun käytetään em perseptroneja diskriminanttifunktiona, voidaan piirreavaruus jakaa lineaarisesti kahteen osaan

Yksittäinen perseptroni suoriutuu siis melko yksinkertaisesta luokitteluongelmasta

(Perseptronin opettamisesta puhuttiin lineaaristen luokittimien yhteydessä)

(Aktivaatiofunktiot voivat olla myös RBF-tyyppisiä)

8.2 MLP-verkko

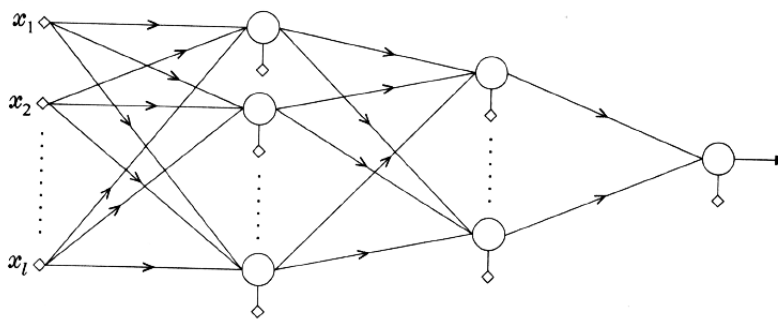
Kytkemällä perseptroneja toisiinsa, saadaan aikaiseksi monimutkaisemmista ongelmista suoriutuva neuroverkko

Eräs tyypillinen ratkaisu on kytkeä perseptronit toisiinsa siten, että ei synny silmukoita ('Feedforward network')

Tämän verkkoarkkitehtuurin yksi yleinen erikoistapaus on monikerrosperseptroni ('Multi-layer perceptron', MLP)

MLP-verkossa perseptronit on järjestetty useiksi kerroksiksi, joiden sisällä ei ole kytkentöjä. Kerrokset on järjestetty ja jokainen kerros kytketty täydellisesti edeltäjäänsä ja seuraajaansa

Kuva 3-kerrosperseptronista, jolla on yksi ulostuloperseptroni:



MLP:n 1. kerros on sisääntulokerros, jossa on jokaiselle piirteelle oma perseptroni. Sisääntulokerroksen perseptronit eivät suorita lainkaan laskentaa

Viimeinen kerros on ulostulokerros, jonka perseptronien ulostulo on koko neuroverkon ulostulo eli vaste. Ulostuloperseptronien aktivaatiofunktio on usein lineaarinen

Ulostuloperseptronien lkm riippuu ongelmasta

Väliin jääviä kerroksia kutsutaan piilokerroksiksi. Piiloperseptronien aktivaatiofunktion tyyppi on usein kaikille sama (jos tyyppi on RBF, verkkoa kutsutaan RBF-verkoksi!)

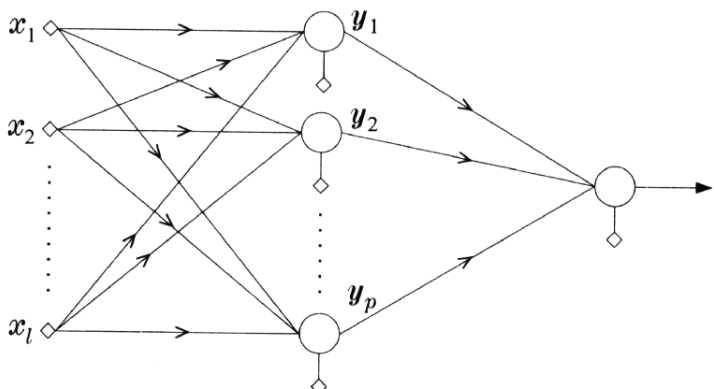
Sopiva piiloperseptronien lkm riippuu ongelmasta

Tarkastellaan seuraavaksi 2 ja 3 kerroksisia MLP-verkkoja, joiden perseptronit on McCulloch-Pitts-tyyppisiä

2-kerrosperseptroni

('Two-layer perceptron')

Yksinkertaisimmillaan MLP-verkossa on vain yksi piilokerros ja yksi ulostuloperseptroni (vastaa kahden luokan ongelmaa)



Seuraava tarkastelu on helppo yleistää useammalle ulostuloperseptronille (useamman kuin kahden luokan ongelma)

Voidaan ajatella, että 2-kerrosperseptroni ratkaisee luokitteluongelman kahdessa perättäisessä vaiheessa

Piilokerros määrittää piirreavaruuteen hypertasoja ja laskee sisääntulon aseman niihin nähden

Ulostulokerros yhdistelee piilokerroksen perseptronien ulostuloja ja päättelee mihin osaan piirreavaruutta ja mihin luokkaan havainto kuuluu

Voidaan myös ajatella, että piilokerros suorittaa epälineaarisen kuvauksen uuteen piirreavaruuteen ja ulostulokerros suorittaa varsinaisen lineaarisen luokittelun (vrt viime luonnolla esitelty yleistetty lineaarinen luokitin)

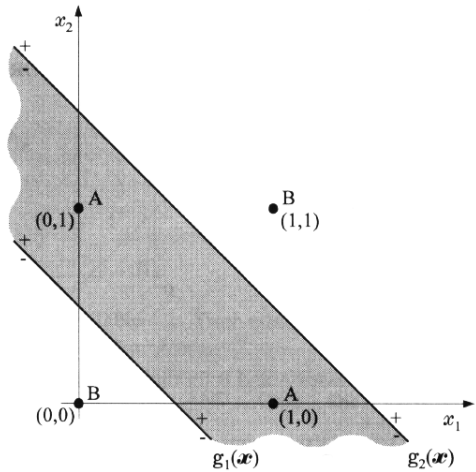
Esimerkki: XOR ja 2-kerrosperseptroni

Ratkaistaan 2-ulotteinen XOR-ongelma käyttäen 2-kerrosperseptronia

Perseptronien aktivaatiofunktiot saavat joko arvon 0 (epätosi) tai 1 (tosi)

Käytetään piilokerroksessa kahta perseptronia

Valitaan piiloperseptronien painokertoimet siten, että tosi-luokan havainnot jäävät perseptronien määrittämien suorien väliin ja epätosi-luokan havainnot taas niiden ulkopuolelle



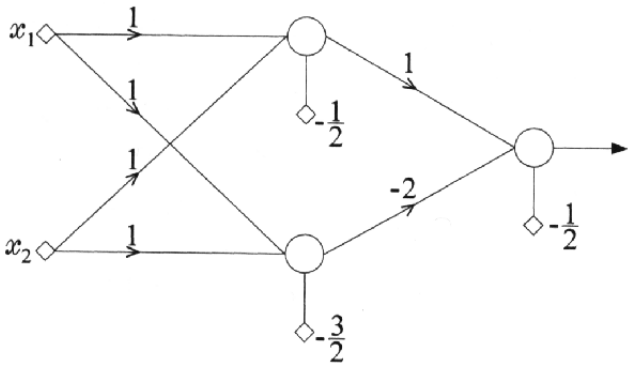
Valitaan ainoan ulostuloperseptronin painokertoimet siten, että verkon ulostulo on 0 tai 1 seuraavan taulukon mukaisesti:

x_1	x_2	y_1^h	y_2^h	y^o
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

missä x_1 ja x_2 ovat verkon sisääntulot, y_1^h ja y_2^h piiloperseptronien ulostulot ja y^o verkon ulostulo

Katsomalla taulukkoa tarkemmin, nähdään helposti, että toinen piiloperseptroni suorittaa AND- ja toinen OR-operaation (vrt aikaisempi esimerkki)

Verkon painokertoimet voidaan valita esimerkiksi seuraavasti:



2-kerrosperseptronin luokittelukyky

Tarkastellaan seuraavaksi millaisista kahden luokan ongelmista 2-kerrosperseptroni selviytyy (tulokset yleistyvät helposti useammalle kuin kahdelle luokalle)

Ol., että aktivaatiofunktiot ovat kaksiarvoisia (0 tai 1)

Ol., että verkossa on l inputperseptronia, p piiloperseptronia ja yksi ulostuloperseptroni

Piilokerros kuvaa piirrevektorit p-ulotteisen hyperkuution H_p kulmiin:

$$H_p = \{[y_1, \ldots, y_p]^T, y_i \in [0,1], 1 \leq i \leq p\} \tag{128}$$

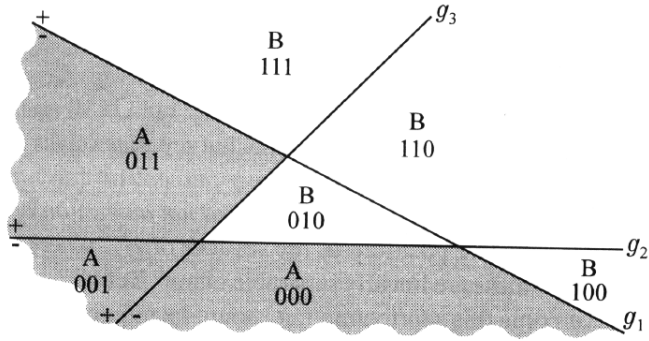
Hyperkuution kulmia ovat ne pisteet, joissa y_i:iden arvot ovat joko nollia tai ykkösiä

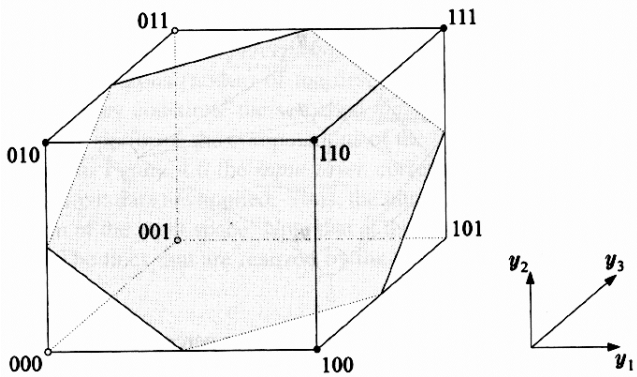
Jokainen piiloperseptroni vastaa hypertasoa. Nämä hypertasot jakavat piirreavaruuden osiin, monitahokkaiksi, joista jokainen vastaa yhtä hyperkuution H_p kulmaa

Ulostuloperseptroni jakaa hyperkuution H_p lineaarisesti kahteen osaan. Eri hyperkuution osiin jäävät kulmat vastaavat eri luokkia

2-kerrosperseptronin muodostamat päätösalueet ovat siis monitahokkaiden unioneita

Päätösalueet eivät voi olla mitä tahansa monitahokkaiden unioneita, koska ulostuloperseptronin tekemä päätös on lineaarinen




011
111
010
110
001
101
000
100
y_2
y_3
y_1

3-kerrosperseptroni

('Three-layer perceptron')

Tarkastellaan seuraavaksi 3-kerrosperseptronia, jossa on 2 piilokerrosta ja joka pystyy suorittamaan monimutkaisemman kuvauksen kuin 2-kerrosperseptroni

Tarkastellaan kahden luokan ongelmaa

Ol. että perseptronien ulostulot ovat kaksiarvoisia

Ol., että luokkaan ω_1 kuuluu kaikki havainnot, jotka sattuvat tiettyihin (J kpl) 1.:n piilokerroksen p:n perseptronin määrittämiin monitahokkaisiin. Luokkaa ω_2 vastaavat kaikki loput monitahokkaat

2. piilokerros pystyy muodostamaan kaikki mahdolliset J:n monitahokkaiden unionit, jos siinä on J piiloperseptronia

Todistus:

- 1. piilokerroksen perseptronit määrittävät p hypertasoa piirreavaruudessa
- 1. piilokerros kuvaa piirrevektorit p-ulotteisen hyperkuution kulmille. Jokainen kulma vastaa yhtä hypertasojen rajaamaa monitahokasta piirreavaruudessa
- Valitaan 2.:n piilokerroksen perseptronien painot siten, että yksi perseptroni erottaa yhden hyperkuution kulman kaikista muista
- Valitaan eroteltavat kulmat siten, että ne vastaavat luokkaa ω_1
- Valitaan ulostuloperseptronin painot siten, että perseptroni toteuttaa OR-operaation
- Nyt verkon ulostulo on tosi (1), jos piirrevektori sattuu johonkin luokkaa ω_1 vastaavaan monitahokkaaseen ja muulloin epätosi (0)

2. piilokerrokseen ei välttämättä tarvita J:tä piiloperseptronia - minimimäärä riippuu ongelmasta

Yhteenveto: 1. piilokerros määrittää hypertasot, 2. piilokerros monitahokkaat ja ulostulokerros eri luokkia vastaavat päätösalueet

(Jos perseptronien aktivaatiofunktiot eivät ole askelfunktioita, MLP-verkon toiminnan tulkinta ei ole näin suoraviivaista!)

8.3 Universaali approksimaattori

Edellisten tarkastelujen perusteella MLP-verkko on selvästi epälineaarinen luokittelumenetelmä

MLP-verkko, jonka piiloperseptronit ovat epälineaarisia ja ulostuloperseptronit lineaarisia ja jolla on yksi ulostuloperseptroni, on yleistetty lineaarinen luokitin (ks edellinen luento)

Tarkastellaan seuraavaksi tällaisen MLP-verkon ja muiden yleistettyjen lineaaristen luokittimien approksimointikykyä

Voidaan ajatella, että yleistetty lineaarinen luokitin approksimoi diskriminanttifunktioita (tai luokkien *a posterior* tnjakaumia) epälineaaristen kantafunktioiden painotetulla summalla

Lähdetään aluksi liikkeelle **Weierstrassin teoreemasta**:

- Olkoot $g(\mathbf{x})$ jatkuva funktio, joka on määritelty piirreavaruuden (suljetussa) kompaktissa osajoukossa S
- Olkoot $\epsilon > 0$ jokin mielivaltaisen pieni vakio
- On olemassa $r(\epsilon)$ ja astetta r oleva polynomi $\Phi(\mathbf{x})$ siten, että

$$|g(\mathbf{x}) - \Phi(\mathbf{x})| < \epsilon, \ \forall \mathbf{x} \in S \tag{129}$$

- Ts, jatkuvaa funktiota voidaan approksimoida mielivaltaisen tarkasti polynomilla (suljetussa) kompaktissa alueessa, kunhan polynomi on riittävän korkeaa astetta

- Approksimointivirhe pienenee säännön $O(\frac{1}{r^{2/l}})$ ('order of magnitude') mukaisesti

Suurin ongelma polynomilla approksimoinnissa on se, että usein riittävän tarkkuuden saavuttamiseksi tarvitaan korkea asteluku r

Korkeaulotteisten polynomien ongelmana on laskennan vaativuuden lisäksi huono yleistämiskyky ja mittausvirheiden kertautuminen

Ongelmaa voidaan kiertää approksimoimalla funktiota paloittain polynomeilla

Hidas approksimointivirheen pienentyminen on ongelma myös muille yleistetyille lineaarisille luokittimille, joiden kantafunktiot ovat kiinteitä

Mikäli kantafunktiot eivät ole kiinteitä vaan määritelty opetusjoukon perusteella säädettävien parametrien avulla (vrt MLP, RBF), approksimaatiovirhe pienenee paljon nopeammin

Tarkastellaan sitten 2-kerrosperseptronia, jonka ainoan ulostuloperseptronin

vaste voidaan kirjoittaa seuraavasti:

$$\Phi(\mathbf{x}) = \sum_{j=1}^{k} w_j^o f(\mathbf{x}^T \mathbf{w}_j^h + w_{j0}^h) + w_0^o \tag{130}$$

missä k on piiloperseptronien lkm ja painokertoimen yläindeksit o ja h viittaavat piilo- ja ulostulokerroksiin

Kun lisäksi oletetaan, että $f(\cdot)$ on litistysfunktio, on vasteella $\Phi(\mathbf{x})$ seuraavat **universaalit approksimointiominaisuudet**:

- Olkoot $g(\mathbf{x})$, jatkuva funktio, joka on määritelty piirreavaruuden kompaktissa osajoukossa S
- Olkoot $\epsilon > 0$ jokin mielivaltaisen pieni vakio
- On olemassa $k(\epsilon)$ ja k:n piiloperseptronin 2-kerrosperseptroni siten, että

$$|g(\mathbf{x}) - \Phi(\mathbf{x})| < \epsilon, \ \forall \mathbf{x} \in S \tag{131}$$

- Approksimointivirhe pienenee säännön $O(\frac{1}{k})$ mukaisesti (Ei riipu piirreavaruuden ulottuvuudesta!)
- Tämä tulos pätee myös RBF-verkolle
- Ts jatkuvaa funktiota voidaan approksimoida mielivaltaisen tarkasti MLP- tai RBF-verkolla kompaktissa alueessa, kunhan perseptroneja (kantafunktiota) on riittävän monta

MLP-approksimoinnin haittapuolena on se, että sopivien painokertoimien valinta on epälineaarinen optimointiongelma (lokaalit minimit!)

Mitä hyötyä on käyttää MLP-verkkoa, jossa on useampi kuin yksi piilokerros? Silloin approksimointi on tehokkaampaa ja tietyn tarkkuuden saavuttamiseksi tarvitaan yleensä vähemmän perseptroneja

8.4 Neuroverkon opettaminen

MLP-verkot, kuten muutkin neuroverkot tai yksittäiset perseptronit, opetetaan opetusjoukon avulla

Opettamisella tarkoitetaan yleensä sopivien synaptisten painojen ja kynnysarvojen (verkon vapaiden parametrien) määräämistä

Laajemmin ajateltuna oppimisena voidaan pitää myös verkon struktuurin valintaa

Oppiminen voi olla joko ohjattua (esim. Backpropagation), ohjaamatonta (esim. Hebbian, Self-Organizing Map eli SOM) tai vahvistettua ('reinforcement learning')

Ohjaamattomassa yritetään löytää havainnoille mielekäs vaste:

- (anti)Hebbian-oppiminen vahvistaa niiden neuronien välisiä kytkentöjä, jotka aktivoituvat yhtä(eri)aikaisesti. Hebbian-tyyppisiä synapseja kutsutaankin korrelaatiosynapseiksi

- On olemassa fysiologisia todisteita sille, että hippocampuksessa, jolla on tärkeä rooli oppimisessa ja muistamisessa, tapahtuu Hebbian-tyyppistä oppimista
- SOM yrittää mallintaa havaintojen rakennetta neuronihilan avulla siten, että lähellä toisiaan olevat neuronit aktivoituvat samankaltaisista havainnoista (Tästä lisää klusteroinnin yhteydessä!)

Ohjatussa oppimisessa haluttu verkon vaste (luokka) tunnetaan. Verkon struktuuri ja synaptiset painot pyritään valitsemaan siten, että verkon todellinen vaste on mahdollisimman lähellä sen haluttua vastetta (ja että verkon struktuuri on mahdollisimman yksinkertainen!)

Ohjatun oppimisen menetelmät voidaan jakaa kahteen luokkaan: 1) opetusjoukon täydelliseen luokitteluun perustuvat menetelmät ja 2) verkon halutun ja todellisen vasteen eroa kuvastavan kustannusfunktion minimointiin perustuvat menetelmät

Opetusjoukon täydelliseen luokitteluun perustuvat menetelmät

Näissä menetelmissä lähdetään liikkeelle yksinkertaisesta neuroverkosta, joka ei yleensä pysty ratkaisemaan annettua tehtävää

Verkkoon lisätään perseptroneja, kunnes se hallitsee opetusjoukon täydellisesti

Lähtökohtana näissä menetelmissä on ajatus ongelman jakamisesta helpompiin osaongelmiin, jotka pystytään ratkaisemaan yhdellä perseptronilla tai hyvin yksinkertaisella verkolla ('constructive techniques')

Verkon parametrit voidaan määrätä lineaaristen luokittimien yhteydessä esitetyillä menetelmillä (esim. perseptroni- tai LMS-algoritmillä)

Verkon asteittaiseen kasvattamiseen on kehitelty useita menetelmiä, jotka perustuvat uusien piilokerrosten tai piiloperseptronien lisäämiseen. Osa menetelmistä tuottaa MLP-verkkoja, mutta osa sallii kytkennät muidenkin kuin perättäisten kerrosten välillä tai kytkennät kerrosten sisällä

Esimerkki: Tiling-algoritmi

Eräs konstruktiivinen menetelmä

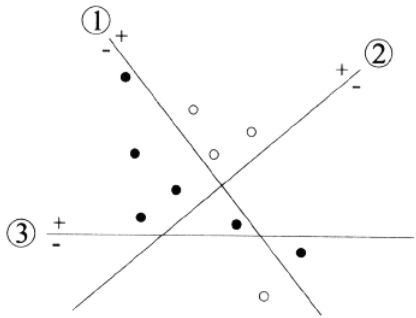
Tuottaa MLP-verkkoja, joissa on useita kerroksia, yleensä vähintään kolme

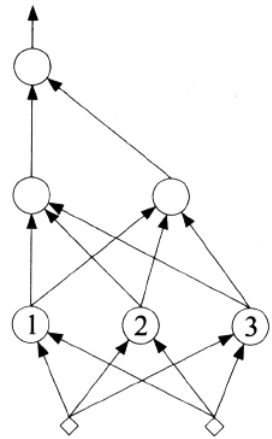
Tarkastellaan kahden luokan ongelmaa:

- Algoritmi lähtee liikkeelle yhdestä master-perseptronista, joka on ensimmäisessä kerroksessa ja joka opetetaan pocket-algoritmillä
- Jaetaan opetusjoukko X kahteen osajoukkoon: päätöspinnan positiiviselle (X^+) ja negatiiviselle puolelle (X^-) sijoittuvat havainnot
- Mikäli jompikumpi tai molemmat osajoukot sisältävät näytteitä molemmista luokista, lisätään ensimmäiseen kerrokseen apu-perseptronit: $n(X^+)$ ja/tai $n(X^-)$, jotka opetetaan osajoukoilla X^+ ja/tai X^-
- Jaetaan osajoukot X^+ ja X^- tarvittaessa uusiksi osajoukoiksi X^{++}, X^{+-}, X^{-+}, ja X^{--} ja lisätään niitä vastaavat apu-perseptronit

- Lisätään apu-perseptroneja kunnes ei pystytä muodostamaan uusia osa-joukkoja
- Lisätään uusi kerros ja siihen master-perseptroni, joka saa syötteensä kaikilta edellisen kerroksen perseptroneilta ja jota opetetaan koko opetusjoukolla. Kasvatetaan uutta kerrosta apu-perseptroneilla kuten ensimmäistä kerrosta
- Lisätään uusia kerroksia, kunnes pystytään luokittelemaan kaikki opetusnäytteet oikein
- Voidaan osoittaa, että verkkoon lisätään vain äärellinen määrä kerroksia (jokainen uusi kerros pystyy luokittelemaan oikein kaikki ne näytteet kuin edellisen kerroksen master-perseptronikin ja vähintään yhden näytteen enemmän)

Luokitteluongelman ratkaisu Tiling-algoritmillä:





Esimerkki: lähimmän naapurin menetelmään perustuva verkko

Toinen esimerkki konstruktiivisesta oppimismenetelmästä on MLP-verkon rakentaminen lähimmän naapurin menetelmään perustuen:

- MLP-verkon 1. piilokerros koostuu perseptroneista, jotka määrittävät hypertason jokaisen opetusnäyteparin välille
- 2. piilokerros muodostaa hypertasoista monitahokkaita AND-perseptronien avulla
- Ulostulokerros muodostaa päätösalueet OR-perseptronien avulla

Menetelmän haittapuolena on se, että verkkoon tarvitaan runsaasti neuroneja

Backpropagation-algoritmi

Seuraavaksi käydään läpi verkon halutun ja todellisen vasteen eroa kuvastavan kustannusfunktion minimointiin perustuvat menetelmä Backpropagation (BP), jossa MLP-verkon struktuuri on kiinnitetty etukäteen

BP-algoritmi on yleisimmin käytetty MLP-verkon opetusmenetelmä

Koska BP-menetelmä perustuu optimointiin, kustannusfunktion on hyvä olla differentioituva

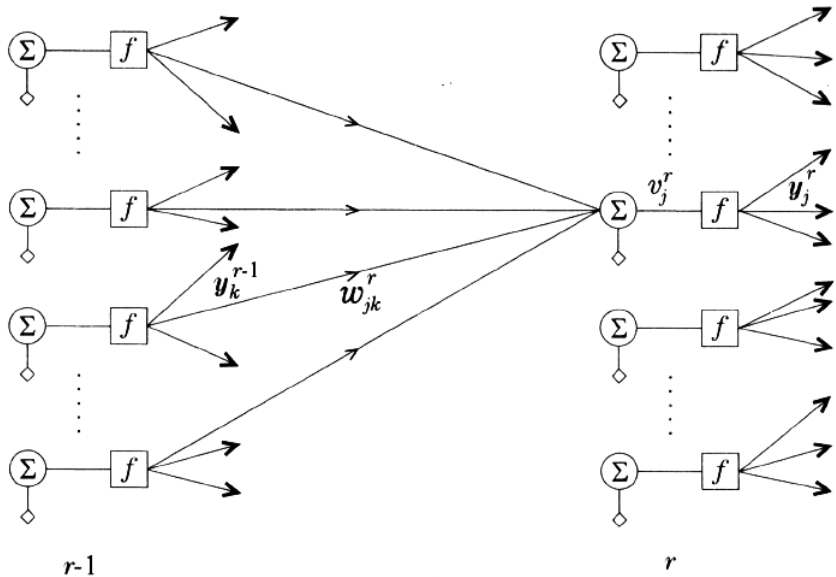
Edellisen perusteella kannataa käyttää jatkuvia ja derivoituvia aktivaatiofunktiota askelfunktioden sijasta

(Tällöin MLP-verkon 1. piilokerros *ei* kuvaa piirrevektoreita hyperkuution kulmiin!)

Käytetetään seuraavia merkintöjä:

- Verkon suorittaman luokittelun onnistumista mitataan kustannusfunktiolla J
- Verkossa on sisääntulokerroksen lisäksi L kerrosta perseptroneja
- Sisääntulokerroksessa on $k_0 = l$ perseptronia, r.:ssä kerroksessa on k_r perseptronia, $1 \leq r \leq L$
- Kaikilla perseptroneilla on samanlainen aktivaatiofunktio (paitsi tietysti sisääntuloperseptroneilla)
- Opetusnäytteitä on käytettävissä N kpl:tta ja niitä vastaavat halutut vasteet tunnetaan: $(\mathbf{y}(i), \mathbf{x}(i)),\ 1 \leq i \leq N$
- Verkon todellinen vaste i.:lle opetusnäytteelle on $\hat{\mathbf{y}}(i)$
- Vektori $\mathbf{w}_j^r$ koostuu r.:n kerroksen j.:n perseptronin synaptisista painoista ja kynnysarvosta

- v_j^r on r.:n kerroksen j.:n perseptronin sisääntulon ja painovektorin $\mathbf{w}_j^r$ sisätulo



BP-menetelmässä verkon parametrejä päivitetään iteratiivisesti 'gradient descent'-menetelmällä:

$$\mathbf{w}_j^r(\text{new}) = \mathbf{w}_j^r(\text{old}) + \Delta\mathbf{w}_j^r, \tag{132}$$

missä

$$\Delta\mathbf{w}_j^r = -\mu\frac{\partial J}{\partial\mathbf{w}_j^r}|_{\mathbf{w}_j^r(\text{old})} \tag{133}$$

ja μ on positiivinen opetusparametri

Gradienttien laskenta

Tyypillisesti BP:n kustannusfunktion on seuraavaa muotoa:

$$J = \sum_{i=1}^{N} E(i), \tag{134}$$

missä $E(i)$ on jokin $\mathbf{y}(i)$:stä ja $\hat{\mathbf{y}}(i)$:stä (verkon haluttu ja todellinen vaste) riippuva funktio, esim. neliövirheiden summa

BP:n päivityssäännössä (132) tarvittava gradientti voidaan laskea silloin seuraavasti:

- $y_k^{r-1}(i)$ on $(r-1)$.:n kerroksen k:n perseptronin i.:ttä opetusnäytettä vastaava ulostulo
- w_{jk}^r on synaptinen painokerroin $(r-1)$.:n kerroksen k.:n perseptronin ja r.:n kerroksen j.:n perseptronin välillä (ks edellinen kuva)
- r.:n kerroksen j.:n perseptronin aktivaatiofunktion $f(\cdot)$:n argumentti on silloin:

$$\begin{aligned} v_j^r(i) &= \sum_{k=1}^{k_{r-1}} w_{jk}^r y_k^{r-1}(i) + w_{j0}^r \\ &= \sum_{k=0}^{k_{r-1}} w_{jk}^r y_k^{r-1}(i), \end{aligned} \tag{135}$$

missä $y_0^r(i) = 1,\ \forall r, i$

- Ulostulokerroksessa $r = L,\ y_k^r(i) = \hat{y}_k(i),\ k = 1, \ldots, k_L$
- Sisääntulokerroksessa $r = 1,\ y_k^{r-1}(i) = x_k(i),\ k = 1, \ldots, k_0$
- Kaavan (135) ja ketjusäännön perusteella:

$$\frac{\partial E(i)}{\partial \mathbf{w}_j^r} = \frac{\partial E(i)}{\partial v_j^r(i)} \frac{\partial v_j^r(i)}{\partial \mathbf{w}_j^r} \tag{136}$$

- Kaavan (135) perusteella:

$$\frac{\partial v_j^r(i)}{\partial \mathbf{w}_j^r} = (\frac{\partial}{\partial w_{j0}^r}, \ldots, \frac{\partial}{\partial w_{jk_{r-1}}^r})^T v_j^r(i) = \mathbf{y}^{r-1}(i), \tag{137}$$

  missä $\mathbf{y}^{r-1}(i) = (1, y_1^{r-1}(i), \ldots, y_{k_{r-1}}^{r-1}(i))^T$
- Käytetään seuraavaa merkintää:

$$\frac{\partial E(i)}{\partial v_j^r(i)} = \delta_j^r(i) \tag{138}$$

  (delta-termi)

- Silloin

$$\Delta \mathbf{w}_j^r = -\mu \sum_{i=1}^{N} \delta_j^r(i) \mathbf{y}^{r-1}(i) \tag{139}$$

Delta-termin laskenta

Edellinen tarkastelu yleistyy siis kaikille muotoa (134) oleville kustannusfunktioille

Seuraavaksi esitetään delta-termin laskenta virheiden neliösummiin perustuvalle kustannusfunktiolle:

$$\begin{aligned} J =& \frac{1}{2} \sum_{i=1}^{N} \sum_{m=1}^{k_L} e_m^2(i) \\ =& \frac{1}{2} \sum_{i=1}^{N} \sum_{m=1}^{k_L} (y_m(i) - \hat{y}_m(i))^2 \end{aligned} \tag{140}$$

Delta-termit lasketaan lähtien liikkeelle ulostuloskerroksesta ja peruuttamalla

(backpropagation!) kohti sisääntulokerrosta:

- Ulostulokerros, $r = L$:

$$\delta_j^L(i) = \frac{\partial E(i)}{\partial v_j^L(i)} \tag{141}$$

$$E(i) = \frac{1}{2}\sum_{m=1}^{k_L} e_m^2(i) = \frac{1}{2}\sum_{m=1}^{k_L}(f(v_m^L(i)) - y_m(i))^2 \tag{142}$$

  Edellisten kaavojen perusteella:

$$\delta_j^L = e_j(i)f'(v_j^L(i)), \tag{143}$$

  missä f' on f:n derivaatta

- Muut kerrokset, $r < L$:

$$\frac{\partial E(i)}{\partial v_j^{r-1}} = \sum_{k=1}^{k_r} \frac{\partial E(i)}{\partial v_k^r(i)} \frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} = \sum_{k=1}^{k_r} \delta_k^r \frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} \tag{144}$$

Käytetään seuraava merkintää:

$$\delta_j^{r-1} = \sum_{k=1}^{k_r} \delta_k^r \frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} \tag{145}$$

Toisaalta

$$\frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} = \frac{\partial \sum_{m=0}^{k_{r-1}} w_{km}^r y_m^{r-1}(i)}{\partial v_j^{r-1}(i)} \tag{146}$$

$$y_m^{r-1} = f(v_m^{r-1}(i)) \tag{147}$$

Edellisten kaavojen perusteella

$$\frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} = w_{kj}^r f'(v_j^{r-1}(i)) \tag{148}$$

Delta-termi voidaan kirjoittaa siis seuraavasti:

$$\delta_j^{r-1}(i) = (\sum_{k=1}^{k_r} \delta_k^r(i) w_{kj}^r) f'(v_j^{r-1}(i)) \tag{149}$$

tai samassa muodossa kuin kaava (143)

$$\delta_j^{r-1}(i) = e_j^{r-1}(i) f'(v_j^{r-1}(i)), \tag{150}$$

missä

$$e_j^{r-1}(i) = \sum_{k=1}^{k_r} \delta_k^r(i) w_{kj}^r \tag{151}$$

Yhteenveto ja huomiota:

Edelliset tarkastelut voidaan koota seuraavaksi algoritmiksi:

- Initialisointi: arvotaan painoille pienet alkuarvot
- Eteenpäinlaskenta: lasketaan jokaiselle opetusnäytteelle $\mathbf{x}(i)$, $i = 1, \ldots, N$,, käyttäen viimeisimpiä painojen arvoja $v_j^r(i)$ ja $y_j^r(i) = f(v_j^r(i))$, $j = 1, \ldots, k_r$, $r = 1, \ldots, L$. Lasketaan lisäksi $E(i)$ ja J
- Taaksepäinlaskenta: lasketaan ulostulokerrokselle $\delta_j^L(i)$ ja järjestyksessä aikaisemmille kerroksille $\delta_j^{r-1}(i)$, $r = L, \ldots, 2$
- Päivitys: lasketaan uudet arvot painoille

$$\mathbf{w}_j^r(\text{new}) = \mathbf{w}_j^r(\text{old}) + \Delta\mathbf{w}_j^r \tag{152}$$

$$\Delta\mathbf{w}_j^r = -\mu \sum_{i=1}^{N} \delta_j^r(i)\mathbf{y}^{r-1}(i) \tag{153}$$

Algoritmin lopetuskriteerinä on usein tavoitearvo J:lle, kynnysarvo J:n muutoksille tai sen gradientin pituudelle, tai maksimimäärä opetuskierroksille ('epochs')

Algoritmin konvergointinopeus riippuu luonnollisesti μ:sta: hyvin pienellä μ:lla konvergointi on varmaa, mutta hidasta; suurella μ:lla vaarana on hyppääminen optimin yli ('overshooting'). Yksi vaihtoehto on käyttää aluksi isompaa ja lopuksi pienempää μ:ta

BP ratkaisee epälineaarisen optimointiongelman, jolla on yleensä useampi kuin yksi lokaali minimi. Ei ole mitään takeita että BP löytäisi niistä parhaimman. Yleensä opetetaan verkko useita kertoja eri alkuarvoilla ja valitaan paras tulos

Edellä painovektoreita päivitettiin kaikkien opetusnäytteiden perusteella ('batch'-versio). Painoja voidaan päivittää myös yksittäisten näytteiden perusteella ('on-line'-versio)

'On-line'-versiossa on enemmän satunnaisuutta, koska kustannusfunktion gradienttien estimaatit perustuvat vain yhteen näytteeseen. Tämä lisätty satun-

naisuus hidastaa konvergointia, mutta auttaa pääsemään pois lokaaleista minimeistä

Satunnaisuutta voidaan lisätä edelleen käyttämällä opetusnäytteet erilaisissa, satunnaisissa järjestyksessä eri opetuskierroksilla

Erilaisia kustannusfunktiota

Ennustusvirheiden neliösumma ei ole ainoa vaihtoehto BP:n kustannusfunktioksi

On olemassa joukko paremmin käyttäytyviä ('gradient descent' löytää globaalin minimin, jos sellainen on olemassa) luokittelunongelmaan sopivia funktioita

Seuraavaksi esitellään niistä muutamia

Ol., että verkon halutut ulostulot y_k ovat satunnaismuuttujia, jotka saavat joko arvon 0 tai 1, ja verkon todelliset ulostulot $\hat{y}_k$ ovat näiden *a posterior*

tn:iä

Silloin kustannusfunktiona voidaan käyttää **'cross-entropy'-funktiota**

$$J = -\sum_{i=1}^{N}\sum_{k=1}^{k_L}(y_k(i)\ln\hat{y}_k(i) + (1-y_k(i))\ln(1-\hat{y}_k(i))) \tag{154}$$

J saa minimiarvonsa, kun $y_k(i) = \hat{y}_k(i)$

Ol., että ulostuloperseptronit ovat toisistaan riippumattomia ja että $\hat{y}_k$ (tai $1-\hat{y}_k$) on tn sille, että k:n ulostuloperseptronin vaste on 1 (tai 0)

Silloin yhden opetusnäyteen vasteelle saadaan seuraava tn:

$$p(\mathbf{y}) = \prod_{k=1}^{k_L}(\hat{y}_k)^{y_k}(1-\hat{y}_k)^{1-y_k} \tag{155}$$

Edellä esitelty J saadaan kun lasketaan koko opetusjoukon vasteiden neg. 'loglikelihood'-funktio

Jos y_k:t ovat todellisia tn:iä välillä $(0, 1)$ ja J:stä vähennetään sen minimiarvo, saadaan

$$J = -\sum_{i=1}^{N}\sum_{k=1}^{k_L}(y_k(i)\ln\frac{\hat{y}_k(i)}{y_k(i)} + (1-y_k(i))\ln\frac{1-\hat{y}_k(i)}{1-y_k(i)}), \qquad (156)$$

missä $0\ln 0 = 0$ binaariarvoisille y_k:ille

Voidaan osoittaa, että 'cross-entropy'-funktio riippuu suhteellisista virheistä, ja, toisin kuin ennustusvirheiden neliösumma, painottaa isoja ja pieniä virheitä yhtä paljon.

Kun y_k:t ovat 0- tai 1-arvoisia ja painot ovat 'cross-entropy'-kustannusfunktion mielessä optimaaliset, $\hat{y}_k$:t ovat todella estimaatteja luokkien *a posteriori* tn:lle $P(\omega_k|\mathbf{x})$

(Edellinen tulos pätee myös ennustusvirheiden neliösummaan perustuvalle kustannusfunktiolle)

'Cross-entropy'-funktioon perustuvan kustannusfunktion selvin etu on se, että se divergoi, jos jokin ulostuloista lähestyy väärää lokaalia optimia ja 'gradient descent'-menetelmä pystyy reagoimaan nopeasti. (Vastaavassa tilanteessa virheiden neliösummaan perustuva kustannusfunktio lähestyisi vakiota)

Eräs vaihtoehto kustannusfunktioksi on luokkien todellisten ja estimoitujen *a posterior* tnjakaumiksi tulkittujen $\mathbf{y}$:n ja $\hat{\mathbf{y}}$:n välinen **Kulback-Liebler-etäisyys**

$$J = -\sum_{i=1}^{N}\sum_{k=1}^{k_L} y_k(i) \ln \frac{\hat{y}_k(i)}{y_k(i)}, \tag{157}$$

joka on aina positiivinen ja saa minimiarvon nolla, kun $\mathbf{y} = \hat{\mathbf{y}}$

Vaikka $\mathbf{y}$ ja $\hat{\mathbf{y}}$ oletettiin tnjakaumiksi, ne eivät välttämättä summaudu ykkösiksi. Tästä ongelmasta päästään eroon käyttämällä ulostulokerroksessa aktivaatiofunktiona softmax-skaalausta:

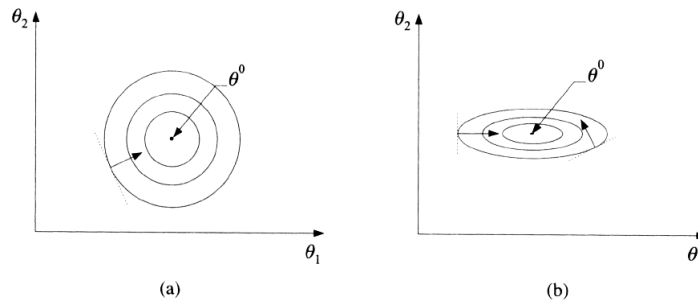
$$\hat{y}_k = \frac{\exp(v_k^L)}{\sum_{k'=1}^{L} \exp(v_{k'}^L)} \tag{158}$$

BP-algoritmin variaatioita

'Gradient descent'-menetelmään perustuvan optimoinnin yleinen ongelma on hidas konvergointi

Konvergointi on hidasta varsinkin silloin, kun gradientin suunta värähtelee opetuskierrosten välillä (kanjoni-efekti, J:n Hessian-matriisin ominaisarvot eivät ole lähellä toisiaan)

Aina kohti optimia osoittava (a) ja värähtelevä (b) gradientti:



Värähtelyä voidaan vähentää **momentti-termin** avulla seuraavasti:

$$\Delta \mathbf{w}_j^r(\text{new}) = \alpha \Delta \mathbf{w}_j^r(\text{old}) - \mu \sum_{i=1}^{N} \delta_j^r(i) \mathbf{y}^{r-1}(i) \tag{159}$$

$$\mathbf{w}_j^r(\text{new}) = \mathbf{w}_j^r(\text{old}) + \Delta \mathbf{w}_j^r(\text{new}) \tag{160}$$

α on positiivinen momenttikerroin, jonka arvot ovat tyypillisesti välillä $0.1 - 0.8$

Momentti-termin keskiarvoistava vaikutus nähdään hyvin seuraavista kaavoista:

$$\Delta \mathbf{w}_j^r(t) = \alpha \Delta \mathbf{w}_j^r(t-1) - \mu \mathbf{g}(t) \tag{161}$$

$$\Delta \mathbf{w}_j^r(T) = -\mu \sum_{t=0}^{T-1} \alpha^t \mathbf{g}(T-t) + \alpha^T \Delta \mathbf{w}_j^r(0), \tag{162}$$

missä $\mathbf{g}(t)$ on J:n gradientti $t.$:llä iteraatioaskeleella. Kun $\alpha < 1$ alkuarvauksen vaikutus lähestyy nollaa muutaman iteraatioaskeleen jälkeen

Oletetaan seuraavaksi, että liikutaan painoavaruudessa alueessa, jossa J:n gradientti on lähestulkoon vakio $\mathbf{g}$ (eli ei värähtelyä!). Silloin

$$\Delta \mathbf{w}_j^r(T) \approx -\mu(1+\alpha+\alpha^2+\dots)\mathbf{g} = -\frac{\mu}{1-\alpha}\mathbf{g} \tag{163}$$

eli käytännössä momenttitermi kasvattaa opetusparametriä ja nopeuttaa konvergointia

Momentti-termin käytön sijasta voidaan käyttää esim. seuraavaan **heuristiikkaan perustuvaa adaptiivistä opetusparametriä**:

$$\begin{aligned}
&\frac{J(t)}{J(t-1)} < 1,\ \mu(t) = r_i\mu(t-1)\ \ \text{(kasvatus)}\\
&\frac{J(t)}{J(t-1)} > c,\ \mu(t) = r_d\mu(t-1)\ \ \text{(piennennys)}\\
&1 \le \frac{J(t)}{J(t-1)} \le c,\ \mu(t) = \mu(t-1),
\end{aligned} \tag{164}$$

missä tyypillisesti $r_i = 1.05$ ja $r_d = 0.7$ ja $c = 1.04$.

Yksi vaihtoehto on käyttää **jokaiselle painolle omaa adaptiivistä opetusparametria**, jota kasvatetaan tai pienennetään J:n derivaatan merkin perusteella. Mikäli derivaatan merkki muuttuu perättäisillä iteraatioaskelilla (värähtelyä?), pienennetään, muuten kasvatetaan opetusparametriä

'Gradient descent'-idean sijasta voidaan käyttää muita optimointitekniikoita, esim. 'Conjugate gradient', 'Newton', 'Kalman filtering', 'Levenberg-Marquard'

Useimmissa em menetelmissä tarvitaan J:n Hessenian-matriisia, joka voidaan laskea samaan tapaan kuin gradientit BP-algoritmissä

8.5 'Weight sharing' - invariantit piirteet

Yksi tärkeimpiä ongelmia hahmontunnistuksessa on erilaisille transformaatioille invarianttien piirteiden löytäminen (esim. käsinkirjoitettujen merkkien tunnistuksessa paikka, koko, kallistus, vinot tai mutkittelevat rivit jne...)

Yksi tapa ratkaista ongelma on suorittaa raakadatalle erilaisia esikäsittelyjä ja piirteille normalisointeja

Toinen tapa on sisällyttää itse luokittelumenetelmään keinoja käsitellä transformaatioita

Neuroverkkojen tapauksessa tämä voidaan toteuttaa rajoittamalla joidenkin painojen arvoja keskenään samoiksi ('weight sharing')

Ns korkeamman asteluvun MLP-verkoissa (**'higher order networks'**) painojen jakamista voidaan soveltaa esim. seuraavasti:

- Perseptronien aktivaatiofunktiot saavat argumenteikseen inputtiensa epälineaarisen kombinaation:

$$f(v) = f(w_0 + \sum_i w_i x_i + \sum_{jk} w_{jk} x_j x_k) \tag{165}$$

- Perseptronin inputtien tulkitaan olevan peräisin 2-ulotteisesta hilasta, jonka pisteet on kytketty toisiinsa suorilla (x_i, x_j)
- Perseptronista saadaan translaatioinvariantti, jos $w_{jk} = w_{rs}$ silloin kun $(x_j, x_k) \parallel (x_r, x_s)$
- ja rotaatioinvariantti, jos $w_{jk} = w_{rs}$ silloin kun $d(x_j, x_k) = d(x_r, x_s)$

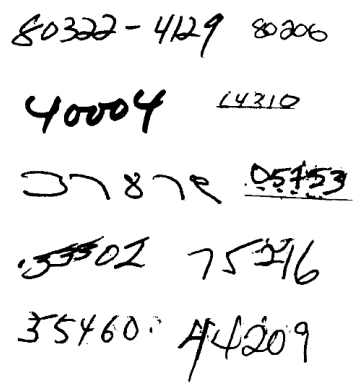
Painojen jakaminen vähentää verkon vapaiden parametrien lkm:ää tuntuvasti ja yleensä verkon oppimiskyky ei heikenny ratkaisevasti

Sovellutusesimerkki painojen jakamisesta:

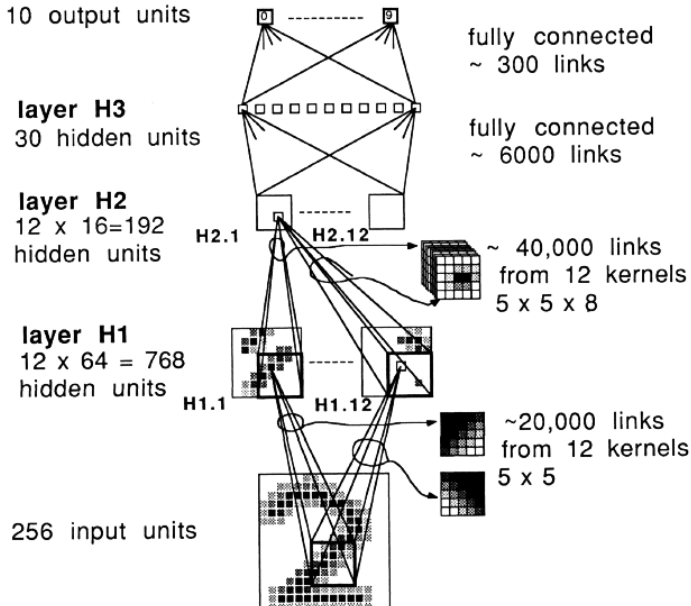
- Backpropagation Applied to Handwritten Zip Code Recognition, LeCun et al, Neural Computation, Vol 1(4), 1989
- Ongelmana on tunnistaa kirjekuorista skannattuja postinumeroita
- Esikäsittelyvaiheessa merkit segmentoidaan ja skaalataan 16×16:n pikselin kokoiksi kuviksi
- Merkit tunnistetaan MLP-verkolla, jossa on 3 piilokerrosta
- Sisääntulokerroksessa on jokaista kuvapikseliä kohti yksi perseptroni
- Ulostulokerroksessa on jokaista luokkaa kohti yksi perseptroni
- 2:n ensimmäisen piilokerroksen perseptroniryhmät toimivat eräänlaisina piirretunnistimina
- Painojen jakamisen ansiosta piirretunnistus on translaatioinvariantia

- Verkon opetus perustuu BP-algoritmiin
- Verkossa on yhteensä $1\,256$ perseptronia ja $64\,660$ kytkentää, mutta vain $9\,760$ vapaata parametriä
- Saavutettu tunnistustarkkuus (virhe noin 5%) on huomattavan korkea

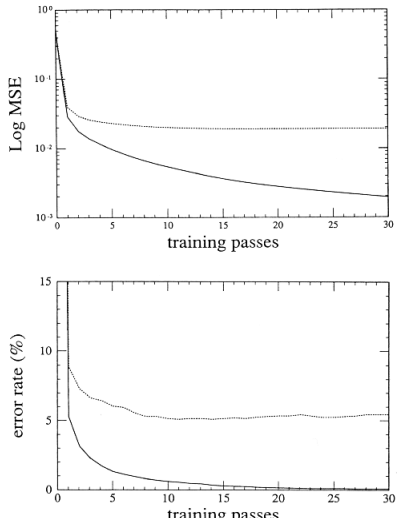
Esimerkkejä tunnistettavista merkeistä:



Kaaviokuva käsikirjoitettuja merkkejä tunnistavasta neuroverkosta:



Ennustusvirheiden neliösumman ja tunnistustarkkuuden kehittyminen opetuksen aikana:



9. SYNTAKTISET JA RAKENTEELLISET MENETELMÄT

(Syntactic Pattern Recognition and Applications, King Sun Fu, Prentice-All, 1982)

Tähän mennessä esitetyissä tunnistusmenetelmissä oletettiin, että hahmot voidaan esittää ja luokitella piirrevektoreiden avulla huomioimatta erityisesti hahmon rakennetta

Joissain tapauksissa tunnistus ei onnistu pelkkien piirrevektoreiden avulla, vaan piirteiden väliset suhteet on myös huomioitava

Piirteiden väliset suhteet voidaan esittää esim. formaalin kielen tai graafin avulla

Syntaktisia ja rakenteellisia menetelmiä voidaan käyttää luokittelun lisäksi hahmojen analysointiin ja generointiin

Syntaktiset menetelmät perustuvat yleensä jäsentimiin ja rakenteelliset esim. graafien vertailuun

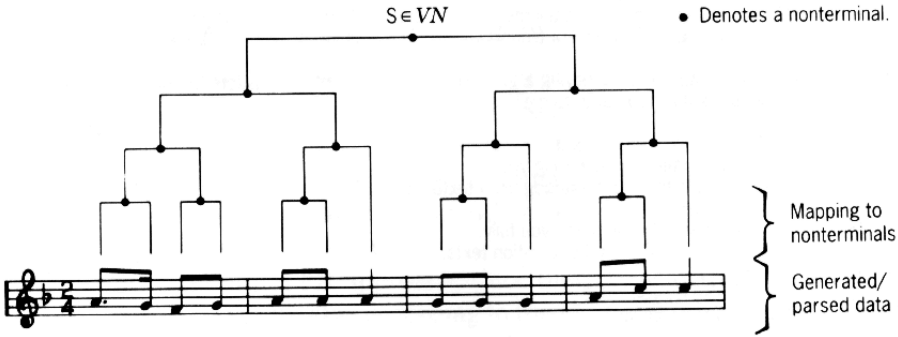
Syntaktisissa ja rakenteellisissa menetelmissä oletetaan usein, että hahmot muodostuvat hierarkisesti yksinkertaisista osista, alihahmoista (esim. kieli - kappaleet - lauseet - sanat - tavut - kirjaimet - vedot)

Tunnistuksessa hahmot jaetaan osiin ja osien väliset suhteet selvitetään

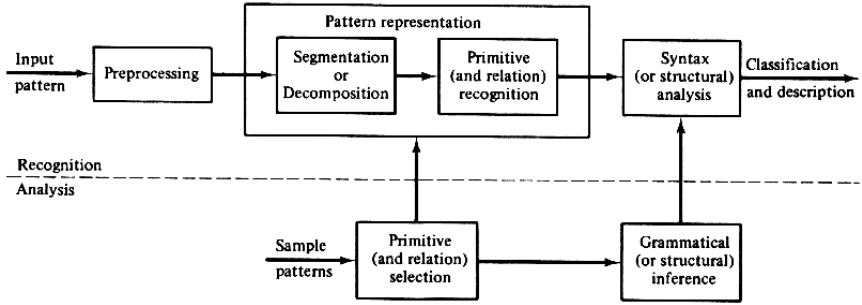
Eri luokissa voi olla samoja alihahmoja, mutta säännöt alihahmojen välisille suhteille ovat erilaiset.

Nämä luokkakohtaiset säännöt saattavat olla ainoa tieto, jonka perusteella hahmot voidaan luokitella

Esimerkki hahmosta, jolla on sisäinen rakenne:



Kaaviokuva syntaktisesta tunnistusjärjestelmästä:



9.1 Formaalit kielet

Voidaan ajatella, että hahmot ovat formaalin kielin sanoja

Formaalin kielen kielioppi kertoo millä tavoin hahmojen perusosasia (symboleja, muuttujia) voidaan yhdistellä suuremmiksi kokonaisuuksiksi (sanoiksi)

Määritellään seuraavaksi **lauserakennekielioppi** ('phrase-structure grammar'):

- Lauserakennekielioppi G koostuu neljästä joukosta $G = (V_N, V_T, P, S)$
- V_N ja V_T ($V_N \cap V_T = \emptyset$) ovat kielen nonterminaali- ja terminaalimuuttujien joukot ja niiden unioni $V_N \cup V_T = V$ on kielen sanasto.
- P on joukko muodostussääntöjä, jotka ovat muotoa $\alpha \rightarrow \beta$, missä α ja β ovat V:n osajoukkoja ja $\alpha \cap V_N \neq \emptyset$
- $S \in V_N$ on aloitusmuuttuja

Joitain hyödyllisiä **merkintöjä**:

- Σ^* on kaikkien äärellisten pituisten symbolijonojen joukko, jotka on muodostettu joukkoon Σ kuuluvista symboleista. λ on tyhjä symbolijono. $\Sigma^+ = \Sigma^* - \{\lambda\}$
- x^n tarkoittaa, että kirjoitetaan symbolijono x n kertaa peräkkäin
- $|x|$ on symbolijonon x pituus
- $\eta \Rightarrow_G \gamma$ tarkoittaa sitä, että symbolijonosta η voidaan *suoraan generoida* symbolijono γ eli $\eta = \omega_1\alpha\omega_2$, $\gamma = \omega_1\beta\omega_2$ ja on olemassa muodostussääntö $\alpha \rightarrow \beta$
- $\eta \Rightarrow^*_G \gamma$ tarkoittaa sitä, että symbolijonosta η voidaan *välillisesti generoida* symbolijono γ eli on olemassa symbolijonojen sarja $\varsigma_1, \ldots, \varsigma_n$ siten, että $\eta = \varsigma_1, \gamma = \varsigma_n$ ja $\varsigma_i \Rightarrow_G \varsigma_{i+1},\ i = 1, \ldots . n-1$. Sarjaa $\varsigma_1, \ldots, \varsigma_n$ voidaan kutsua γ:n johtamiseksi ('derivation') η:sta

Jos G on lauserakennekielioppi, silloin

$$L(G) = \{x | x \in V_T^* \text{ siten, että } S \Rightarrow_G^* x\} \tag{166}$$

on sitä vastaava *lauserakennekieli*

Mikäli kieleen $L(G)$ kuuluva symbolijono voidaan generoida useammalla kuin yhdellä tavalla, kielioppi G on *moniselitteinen* ('ambiguous'). Hahmontunnistussovellutuksissa pyritään yleensä muodostamaan kielioppi, joka on yksiselitteinen (jäsennys on helpompaa!)

Lauserakennekieliopit voidaan jakaa neljään tyyppiryhmään muodostussääntöjen perusteella:

- Tyyppi 0: **rajoittamattomat kieliopit**

  Tällaiset kieliopit ovat liian yleisiä ollakseen hyödyllisiä ja on erittäin vaikea päätellä onko annettu symbolijono kieliopin mukainen

  (Rajoittamatonta kielioppia vastaa rajoittamaton kieli)

- Tyyppi 1: **kontekstiriippuvat kieliopit**

  Kaikki kieliopin muodostussäännöt ovat muotoa $\varsigma_1 A \varsigma_2 \rightarrow \varsigma_1 \beta \varsigma_2$, missä $A \in V_N$, $\varsigma_1, \varsigma_2, \beta \in V^*$ ja $\beta \neq \lambda$

  (Kontekstiriippuvaa kielioppia vastaa kontekstiriippuva kieli)

- Tyyppi 2: **kontekstiriippumattomat kieliopit**

  Kaikki kieliopin muodostussäännöt ovat muotoa $A \rightarrow \beta$, missä $A \in V_N$ ja $\beta \in V^+$

  Kontekstiriippumattomat kieliopit ovat kaikkien vähiten rajoitettu kielioppityyppi, jolle on olemassa tehokkaita jäsennysmenetelmiä

  Muodostussäännöt voidaan esittää myös puun ('derivation / parsing tree') avulla, joka muodostetaan seuraavasti:

  - Puun solmut vastaavat V:hen kuuluvia muuttujia
  - Puun juuri vastaa aloitusmuuttujaa S
  - Mikäli solmulla on jälkeläisiä, se vastaa nonterminaalimuuttujien joukkoon V_N kuuluvaa muuttujaa

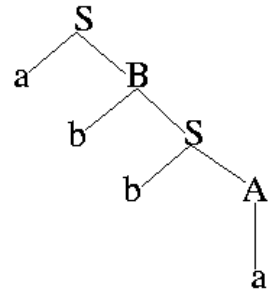
– Mikäli solmut $n_1, \ldots, n_k$ (vastaavat muuttujia $A_1, \ldots, A_k$) ovat solmun n lapsia (vastaa muuttujaa A), silloin on olemassa muodostussääntö $A \to A_1 A_2 \ldots A_k$

Esimerkki: Olkoot $G = (V_N, V_T, P, S)$, missä $V_N = \{S, A, B\}$, $V_T = \{a, b\}$ ja

$$\begin{aligned} P = \{ & S \to aB, && S \to bA, \\ & A \to aS, && A \to bAA, \\ & A \to a, && B \to bS, \\ & B \to aBB, && B \to b\} \end{aligned}$$

Nähdään helposti, että kielioppi G on kontekstiriippumaton. Kielioppia vastaava kieli $L(G)$ koostuu symbolijoinoista, joissa on yhtä monta a:ta ja b:tä ja vähintään yksi kumpaakin

Edellä määriteltyyn kieleen kuuluvaa sanaa $abba$ vastaava puu:



$(S \Rightarrow aB \Rightarrow abS \Rightarrow abbA \Rightarrow abba)$

(Kontekstiriippumatonta kielioppia vastaa kontekstiriippumaton kieli)

- Tyyppi 3: **Säännölliset kieliopit** ('finite-state', 'regular')

  Kaikki kieliopin muodostussäännöt ovat muotoa $A \rightarrow aB$ tai $A \rightarrow b$, missä $A, B \in V_N$ ja $a, b \in V_T$. A, B, a ja b ovat kaikki yksittäisiä symboleja

Säännöllisessä kieliopin muodostussäännön molemmilla puolilla voi siis olla korkeintaan yksi nonterminaalisymboli

Säännöllisen kieliopin hyviä puolia: 1) jäsennys voidaan tehdään äärellisellä tilakoneella, 2) kieliopille voidaan muodostaa yksinkertainen graafinen esitys ja 3) on olemassa menetelmiä, joiden avulla voidaan testata kahden kieliopin, G_1 ja G_2 , *ekvivalenttisuus* (ts $L(G_1) = L(G_2)$)

Esimerkki: Tarkastellaan kielioppia $G = (V_N, V_T, P, S)$, missä $V_N = \{S, A\}$, $V_T = \{a, b\}$ ja $P = \{S \rightarrow aA,\ A \rightarrow aA,\ A \rightarrow b\}$. Nähdään helposti, että kielioppi G on säännöllinen ja sitä vastaava kieli on $L(G) = \{a^n b | n = 1, 2, \dots\}$

(Säännöllistä kielioppia vastaa säännöllinen kieli)

- Jokainen säännöllinen kielioppi on kontekstiriippumaton; jokainen kontekstiriippumaton kielioppi on kontekstiriippuva; jokainen kontekstiriippuva kielioppi on rajoittamaton kielioppi

Säännölliset kieliopit ja äärelliset tilakoneet

Toinen tapa tarkastella kieltä on määritellä siihen kuuluvat sanat tunnistusmenetelmän avulla: vain tunnistusmenetelmän hyväksymät sanat kuuluvat kieleen

Tunnistusmenetelmänä voidaan käyttää esim. tilakonetta

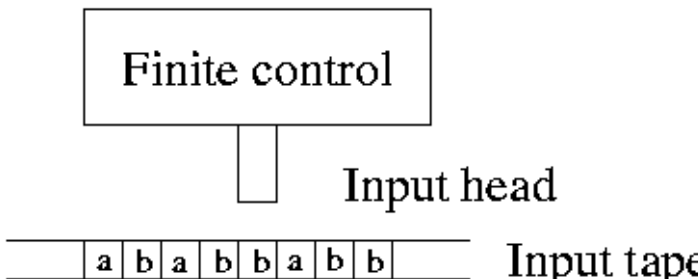
Ääreellinen tilakone ('finite-state automaton') voi hyväksyä vain säännöllisen kieliopin mukaisia sanoja

Määritellään **deterministinen, äärellinen tilakone**:

- Tilakone A koostuu viidestä osasta: $A = (\Sigma, Q, \delta, q_0, F)$
- Σ on äärellinen joukko symboleita, joilla tilakonetta ohjataan
- Q on äärellinen joukko tilakoneen tiloja
- $\delta(q, a) = q'$ ($\delta : Q \times \Sigma \rightarrow Q$) määrittää kuinka tilakone käyttäytyy eli vaihtaa tilaa erilaisilla ohjauksilla. Usein δ esitetään taulukkona

- $q_0 \in Q$ on tilakoneen alkutila
- $F \in Q$ on tilakoneen lopputilojen joukko

Kuva äärellisestä tilakoneesta:



Ol., että äärellinen tilakone A on tilassa $q \in Q$ ja saa ohjaussymbolin $a \in \Sigma$. Silloin tilakoneen tila vaihtuu säännön $\delta(q, a) = q'$ mukaisesti uuteen tilaan q'.

Pidempi sarja tilamuutoksia ja ohjauksia voidaan kirjoittaa käyttäen seuraavia merkintöjä:

$$\delta(q, \lambda) = q,\ \delta(q, xa) = \delta(\delta(q, x), a), \tag{167}$$

missä $x \in \Sigma^*$ ja $a \in \Sigma$

Sanotaan, että tilakone A *hyväksyy* symbolijonon $x \in \Sigma^*$, jos

$$\delta(q_0, x) = p \text{ ja } p \in F \tag{168}$$

Tilakoneen A hyväksymien symbolijonojen joukko on kieli $T(A)$

$$T(A) = \{x | \delta(q_0, x) \in F\} \tag{169}$$

Epädeterministinen äärellinen tilakone on muuten samanlainen kuin edellä esitelty deterministinen äärellinen tilakone paitsi että δ antaa yhden uuden tilan sijasta joukon mahdollisia uusia tiloja eli

$$\delta(q,\lambda) = \{q\},\ \delta(q,x) = \{q_1,\ldots,q_l\},\ \delta(q,xa) = \bigcup_{q_i \in \delta(q,x)} \delta(q_i,a), \quad (170)$$

missä $x \in \Sigma^*$ ja $a \in \Sigma$

Käytetään seuraavaa merkintää:

$$\delta(\{q_1,\ldots,q_l\},x) = \bigcup_{i=1}^{l} \delta(q_i,x) \quad (171)$$

Tilakone A *hyväksyy* symbolijonon $x \in \Sigma^*$, jos

$$p \in \delta(q_0,x) \text{ ja } p \in F \quad (172)$$

Tilakoneen A hyväksymien symbolijonojen joukko on kieli $T(A)$

$$T(A) = \{x | p \in \delta(q_0,x) \text{ ja } p \in F\} \quad (173)$$

Symbolijonojen joukon L hyväksyvää **epädeterminististä tilakonetta A vastaava deterministinen tilakone** A', joka myös hyväksyy joukon L, voidaan muodostaa seuraavasti:

- Halutaan siis, että $L = T(A) = T(A')$
- $A = (\Sigma, Q, \delta, q_0, F)$ ja $A' = (\Sigma', Q', \delta', q_0', F')$
- $\Sigma' = \Sigma$
- A':n tilat Q' ovat A:n tilojen Q kaikki osajoukot
- A':n lopputilat F' ovat ne joukkoon Q':n kuuluvat tilat, joihin kuuluu F:ään kuuluvia tiloja
- Merkitään A':n tiloja seuraavasti: $[q_1, \ldots, q_i] \in Q'$ $(q_1, \ldots, q_i \in Q)$ ja $q_0' = [q_0]$
- $\delta'([q_1, \ldots, q_i], a) = [p_1, \ldots, p_j]$, jos $\delta(\{q_1, \ldots, q_i\}, a) = \bigcup_{k=1}^{i} \delta(q_k, a) = \{p_1, \ldots, p_j\}$

(voidaan puhua yksinkertaisesti vain äärellisistä tilakoneista)

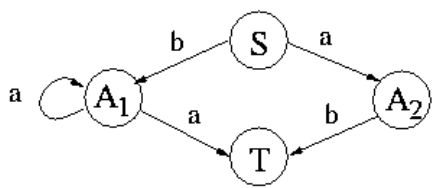
Kielen $L(G)$ **säännöllistä kielioppia** $G = (V_N, V_T, P, S)$ **vastaava äärellinen tilakone** $A = (\Sigma, Q, \delta, q_0, F)$ voidaan muodostaa seuraavasti:

- Halutaan siis, että $T(A) = L(G)$
- $\Sigma = V_T$
- $Q = V_N \cup \{T\}$
- $q_0 = S$
- Jos $(S \to \lambda) \in P$, silloin $F = \{S, T\}$; muuten $F = \{T\}$
- $T \in \delta(B, a)$, jos $(B \to a) \in P$ $(B \in V_N, a \in V_T)$
- $\delta(B, a)$:hen kuuluvat kaikki tilat, joille pätee: $(B \to aC) \in P$ $(C \in V_N)$ ja $\delta(T, a) = \emptyset \; \forall a \in V_T$

Esimerkki: Muodostetaan säännöllistä kielioppia G vastaava äärellinen tilakone A. $G = (V_T, V_N, P, S)$, missä $V_T = \{a, b\}$, $V_N = \{S, A_1, A_2\}$ ja

$$\begin{aligned} P = \{ & S \to aA_2, \\ & S \to bA_1, \\ & A_1 \to a, \\ & A_1 \to aA_1, \\ & A_2 \to b\} \end{aligned}$$

Kielioppia G vastaava tilakone A:



Symbolijonojen joukon $T(A)$ hyväksyvää **äärellistä tilakonetta** A **vastaava säännöllinen kielioppi** G, jonka hyväksymälle kielelle pätee $L(Q) = T(A)$, voidaan muodostaa seuraavasti:

- $V_N = Q$
- $V_T = \Sigma$
- $S = q_0$
- $(B \to aC) \in P$, jos $\delta(B, a) = C$ $(B, C \in Q,\ a \in \Sigma)$
- $(B \to a) \in P$, jos $\delta(B, a) = C$ ja $C \in F$

Esimerkki: Muutetaan säännöllinen kielioppi G epädeterministiseksi äärelliseksi tilakoneeksi A ja siitä edelleen deterministiseksi äärelliseksi tilakoneeksi A'

- Kielioppi on $G = (V_N, V_T, P, S)$, missä $V_N = \{S, B\}$, $V_T = \{a, b\}$ ja

$$\begin{aligned} P = \{&S \to aB, \\ &B \to aB, \\ &B \to bS, \\ &B \to a\} \end{aligned}$$

- G:tä vastaava epädeterministinen äärellinen tilakone $A = (\Sigma, Q, \delta, q_0, F)$ muodostetaan seuraavasti: $\Sigma = V_T = \{a, b\}$, $Q = V_N \cup \{T\} = \{S, B, T\}$, $q_0 = S$, $F = \{T\}$ ja

$$\begin{aligned} &\delta(S, a) = \{B\}, && \text{koska } (S \to aB) \in P \\ &\delta(S, b) = \emptyset, \\ &\delta(B, a) = \{B, T\}, && \text{koska } (B \to aB), (B \to a) \in P \\ &\delta(B, b) = \{S\}, && \text{koska } (B \to bS) \in P \\ &\delta(T, a) = \delta(T, b) = \emptyset \end{aligned}$$

- A:ta vastaava deterministinen äärellinen tilakone on $A' = (\Sigma', Q', \delta', q_0', F')$, missä $\Sigma = \Sigma' = \{a, b\}$, $Q' = \{\emptyset, [S], [B], [T], [S, B], [S, T], [B, T], [S, B, T]\}$, $q_0' = [S]$, $F' = \{[T], [S, T], [B, T], [S, B, T]\}$ ja

$$\begin{aligned} \delta'([S], a) &= [B], & \delta'([S], b) &= \emptyset, \\ \delta'([B], a) &= [B, T], & \delta'([B], b) &= [S], \\ \delta'([B, T], a) &= [B, T], & \delta'([B, T], b) &= [S], \\ \delta'(\emptyset, a) &= \delta'(\emptyset, b) = \emptyset \end{aligned}$$

  (on olemassa myös muita sääntöjä δ', mutta koska A' ei pääse kuin tiloihin $\emptyset$, $[S]$, $[B]$ ja $[B, T]$, voidaan muut tilat ja niitä koskevat säännöt jättää huomioimatta)

- Nyt siis $L(G) = T(A) = T(A')$

Kontekstiriippumattomat kieliopit ja pinomuistitilakoneet

Kontekstiriippumattoman kieliopin mukaisia sanoja voidaan tunnistaa pinomuistitilakoneen ('pushdown automaton') avulla

Kaikki kontekstiriippumattoman kielen $L(G)$ sanat voidaan muodostaa kieliopin $G = (V_N, V_T, P, S)$, jonka muodostussäännöt ovat muotoa $A \to BC$ tai $A \to a$ $(A, B, C \in V_N,\ a \in V_T)$, avulla (*'Chomsky normal form'*)

Kaikki kontekstiriippumattoman kielen $L(G)$ sanat voidaan muodostaa kieliopin $G = (V_N, V_T, P, S)$, jonka muodostussäännöt ovat muotoa $A \to a\alpha$ $(A \in V_N,\ a \in V_T,\ \alpha \in V_N^*)$, avulla (*'Greibach normal form'*)

(on olemassa tunnettuja algoritmeja, joiden avulla kontekstiriippumattomat kieliopit voidaan muuntaa em muotoihin)

Jokaiselle kontekstiriippumattomalle kielelle $L(G)$ on olemassa vakiot p ja q, joille pätee: jos $|z| > p,\ z \in L(G)$, silloin voidaan kirjoittaa $z = uvwxy$, missä $|vwx| \leq q$, v ja x eivät ole yhtäaikaa λ, ja $uv^iwx^iy \in L(G),\ \forall i \geq 1$ (*'pumping lemma'*)

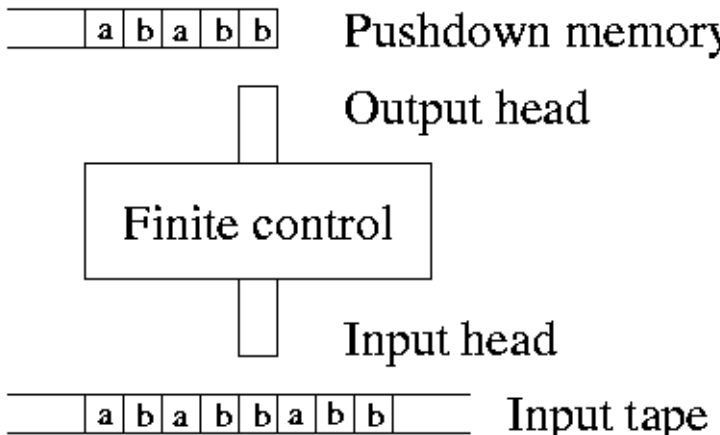
Kontekstiriippumattoman kieliopin $G = (V_N, V_T, P, S)$ sanotaan olevan *rekursiivinen* ('self-embedding'), jos on olemassa $A \in V_N$ siten, että $A \Rightarrow_G^* \alpha_1 A \alpha_2,\ \alpha_1, \alpha_2 \in V^+$ (myös muuttujan A voidaan sanoa olevan rekursiivinen)

Rekursiiviset muuttujat tuotavat muotoa uv^iwx^iy olevia sanoja ja erottelevat kontekstiriippumattomat kieliopit säännöllisistä

Pinomuistitilakone on kuin äärellinen tilakone, jolla on lisäksi mielivaltaisen pitkä pinomuisti. Se määritellään seuraavasti:

- Pinomuistitilakone M koostuu seitsemästä osasta:
  $M = (\Sigma, Q, \Gamma, \delta, q_0, Z_0, F)$
- Σ on äärellinen joukko symboleita, joilla konetta ohjataan
- Q on äärellinen joukko koneen tiloja
- Γ on äärellinen joukko symboleita, joita voidaan tallettaa pinomuistiin
- $q_0 \in Q$ on koneen alkutila
- $Z_0 \in \Gamma$ on pinomuistin alkutila
- $F \in Q$ on koneen lopputilojen joukko
- $\delta : \ Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to Q \times \Gamma^*)$ kertoo kuinka kone käyttäytyy eli mikä on koneen ja muistin uusien mahdollisten tilojen joukko ohjauskomennon jälkeen

Kuva pinomuistitilakoneesta:



Ol., että pinomuistitilakone M on tilassa q, sen pinomuistin päällimmäinen symboli on Z ja se saa ohjaussymbolin a. Silloin pinomuistitilakoneen uudet mahdolliset tilat ja sen pinomuistin päällimmäiset symbolit saadaan seuraavasta säännöstä:

$$\delta(q, a, Z) = \{(q_1, \gamma_1), \ldots, (q_m, \gamma_m)\}, \tag{174}$$

missä $q, q_1, \ldots, q_m \in Q$, $a \in \Sigma$, $Z \in \Gamma$ ja $\gamma_1, \ldots, \gamma_m \in \Gamma^*$ (γ_i siis korvaa Z:n pinomuistissa)

(λ-siirto: $\delta(q, \lambda, Z) = \{(q_1, \gamma_1), \ldots, (q_m, \gamma_m)\}$)

Pinomuistitilakoneen *konfiguraatio* voidaan esittää parin (q, γ) avulla, missä $q \in Q$ on koneen tila ja $\gamma \in \Gamma^*$ on pinomuistin sisältö

Merkintä

$$a : \ (q, Z\gamma) \vdash_M (q', \beta\gamma) \tag{175}$$

tarkoittaa, että sääntöjen δ perusteella ohjaus a vaihtaa pinomuistitilakoneen M konfiguraation $(q, Z\gamma)$:sta $(q', \beta\gamma)$:ksi ($a \in (\Sigma \cup \{\lambda\})$, $\gamma, \beta \in \Gamma^*$ ja $Z \in \Gamma$)

Jos on olemassa ohjaukset $a_1, \ldots, a_n$ $(a_i \in (\Sigma \cup \{\lambda\})\ \forall i = 1, \ldots, n)$, koneen tilat $q_1, \ldots, q_{n+1}$ $(g_l \in Q\ \forall l = 1, \ldots, n+1)$ ja muistisymbolijonot $\gamma_1, \ldots, \gamma_{n+1}$ $(\gamma_j \in \Gamma^*\ \forall j = 1, \ldots, n+1)$ siten, että

$$a_i :\ (q_i, \gamma_i) \vdash_M (q_{i+1}, \gamma_{i+1}),\ \forall i = 1, \ldots, n, \tag{176}$$

voidaan kirjoittaa

$$a_1 a_2 \ldots a_n :\ (q_1, \gamma_1) \vdash_M^* (q_{n+1}, \gamma_{n+1}) \tag{177}$$

Voidaan määritellä kahdella tavalla pinomuistitilakoneen M hyväksymä kieli:

- $T(M) = \{x | x : (q_0, Z_0) \vdash_M^* (q, \gamma),\ \forall \gamma \in \Gamma^*,\ q \in F\}$
  (päädytään alkutilasta lopputilaan)
- $N(M) = \{x | x : (q_0, Z_0) \vdash_M^* (q, \lambda),\ \forall q \in Q\}$ (pinomuisti jää tyhjäksi)

L on $N(M_1)$ jollekin pinomuistitilakoneelle M_1 ainoastaan silloin, kun se on $T(M_2)$ jollekin pinomuistitilakoneelle M_2

Jokaiselle **kontekstiriippumattomalle kielelle** $L(G)$, joka määritelty 'Greibach normal form'-muodossa olevalla kieliopilla $G = (V_N, V_T, P, S)$, **on olemassa vastaava epädeterministinen pinomuistitilakone** $M = (\Sigma, Q, \Gamma, \delta, q_0, Z_0, F)$ siten, että $N(M) = L(G)$ ja

- $\Sigma = V_T$
- $Q = \{q_1\}$
- $\Gamma = V_N$
- $q_0 = q_1$
- $Z_0 = S$
- $F = \emptyset$
- $(q_1, \gamma) \in \delta(q_1, a, A)$, jos $(A \to a\gamma) \in P$

Esimerkki: Muodostetaan kielioppia $G = (V_N, V_T, P, S)$ vastaava epädeterministinen pinomuistitilakone $M = (\Sigma, Q, \Gamma, \delta, q_0, Z_0, F)$

- G on määritetty seuraavasti: $V_N = \{S, A, B\}$, $V_T = \{a, b\}$ ja

$$\begin{aligned} P = \{ & S \to bA, \quad && S \to aB, \\ & A \to a, && B \to b, \\ & A \to aS, && B \to bS, \\ & A \to bAA, && B \to aBB\} \end{aligned}$$

- M muodostetaan seuraavasti: $\Sigma = V_T = \{a, b\}$, $Q = \{q_1\}$, $\Gamma = V_N = \{S, A, B\}$, $q_0 = q_1$, $Z_0 = S$ ja

$$\begin{aligned}
\delta(q_1, a, S) &= \{(q_1, B)\}, && \text{koska } (S \to aB) \in P \\
\delta(q_1, b, S) &= \{(q_1, A)\}, && \text{koska } (S \to bA) \in P \\
\delta(q_1, a, A) &= \{(q_1, S), (q_1, \lambda)\}, && \text{koska } (A \to aS), (A \to a) \in P \\
\delta(q_1, b, A) &= \{(q_1, AA)\}, && \text{koska } (A \to bAA) \in P \\
\delta(q_1, a, B) &= \{(q_1, BB)\}, && \text{koska } (B \to aBB) \in P \\
\delta(q_1, b, B) &= \{(q_1, S), (q_1, \lambda)\}, && \text{koska } (B \to bS), (B \to b) \in P
\end{aligned}$$

- Nyt siis $N(M) = L(G)$

Jokaiselle kielen $N(M)$ hyväksyvälle **pinomuistitilakoneelle** $M = (\Sigma, Q, \Gamma, \delta, q_0, Z_0, F)$ **on olemassa vastaava kontekstiriippumaton kielioppi** $G = (V_N, V_T, P, S)$ siten, että $L(G) = N(M)$ ja

- $V_T = \Sigma$
- V_N on muotoa $[q, A, p]$ $(q, p \in Q,\ A \in \Gamma \cup S)$ olevien triplettien joukko
- $(S \to [q_0, Z_0, q]) \in P\ \forall q \in Q$
- $([q, A, p] \to a[q_1, B_1, q_2][q_2, B_2, q_3] \dots [q_m, B_m, q_{m+1}]) \in P$ kaikille niil-le $q, q_1, \dots, q_{m+1} \in Q,\ p = q_{m+1}$, $a \in (\Sigma \cup \{\lambda\})$ ja $A, B_1, \dots, B_M \in \Gamma$, joille pätee $(q_1, B_1 B_2 \dots B_m) \in \delta(q, a, A)$. (Jos $m = 0$, silloin $q_1 = p$, $(p, \lambda) \in \delta(p, a, A)$ ja $([q, A, p] \to a) \in P$)

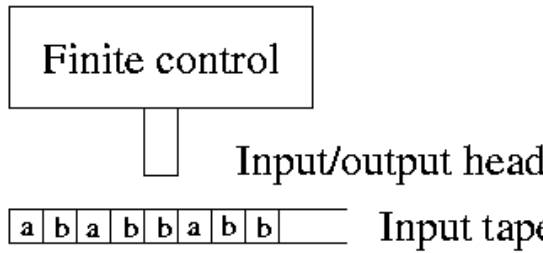
Säännöllisten kielten ja niitä vastaavien äärellisten tilakoneiden tapauksessa deterministinen ja epädeterministinen malli pystyttiin muuntamaan toisikseen. Näin ei kuitenkaan ole kontekstiriippumattomien ja pinomuistitilakoneiden tapauksessa

Rajoittamattomat ja kontekstiriippuvat kieliopit ja Turing-koneet

Turing-kone T on äärellinen tilakone, joka pystyy vaikuttamaan ohjauskomentoihinsa ja se voidaan määritellä seuraavasti:

- Turing-kone koostuu kuudesta osasta: $T = (\Sigma, Q, \Gamma, \delta, q_0, F)$
- Q on äärellinen joukko koneen tiloja
- Γ on äärellinen joukko symboleita, joista yksi on ns tyhjä symboli B
- $\Sigma \in (\Gamma - \{B\})$ on ohjaussymboleiden joukko
- $\delta : \ Q \times \Gamma \rightarrow Q \times (\Gamma - \{B\} \times \{L, R\})$ kertoo kuinka kone käyttäytyy eli mikä on koneen ja ohjausnauhan tila ohjauskomennon jälkeen. L ja R kertovat kumpaan suuntaan ohjausnauhan luku- ja kirjoituspää siirtyy
- q_0 on koneen alkutila
- $F \in Q$ on koneen lopputilojen joukko

Kuva Turing-koneesta:



Turing-koneen konfiguraatio voidaan esittää kolmikon (q, α, i) avulla, missä $q \in Q$ on koneen tila, $\alpha \in \Sigma^*$ ohjausnauhan kirjoitettu osuus ja i luku- ja kirjoituspään sijainti

Merkintä

$$(q, A_1 A_2 \dots A_n, i) \vdash_T (p, A_1 \dots A_{i-1} X A_{i+1} \dots A_n, i \pm 1) \qquad (178)$$

tarkoittaa, että sääntöihin δ perustuen Turing-koneen konfiguraatio voi muuttua $(q, A_1 A_2 \dots A_n, i)$:stä $(p, A_1 \dots A_{i-1} X A_{i+1} \dots A_n, i \pm 1)$:ksi siten, että ohjausnauhan kohtaan i kirjoitetaan symboli $X \in \Sigma$ ja luku- ja kirjoituspää siirtyy yhden askeleen oikealle tai vasemmalle ('elementary move')

Mikäli kaksi Turing-koneen konfiguraatioita voi vaihtua toisikseen äärellisellä määrällä em perusoperaatioita, voidaan käyttää merkintää $\vdash_T^*$ konfiguraatioiden välillä

Turing-koneen hyväksymä kieli voidaan määritellä seuraavasti:

$$\{x | x \in \Sigma^*,\ (q_0, x, 1) \vdash_T^* (q, \alpha, i),\ q \in F,\ \alpha \in \Gamma^*, i \geq 1\} \qquad (179)$$

Rajoittamatonta kielioppia vastaava kieli voidaan määritellä Turing-koneen avulla; Turing-koneen hyväksymä kieli voidaan esittää rajoittamattoman kielioppin avulla

Lineaarisesti rajoitettu tilakone ('linear-bounded automaton') on epädeterministinen Turing-kone, jonka luku- ja kirjoituspää ei voi siirtyä ohjausnauhan kirjoittamattomaan kohtaan

Kontekstiriippuvaa kielioppia vastaava kieli voidaan määritellä lineaarisesti rajoitetun tilakoneen avulla; lineaarisesti rajoitetun tilakoneen hyväksymä kieli on kontekstiriippuva

Ei ole olemassa tehokkaita, yleispäteviä algoritmeja, joiden avulla voitaisiin löytää rajoittamatonta tai kontekstiriippuvaa kieltä vastaava tilakone

9.2 Formaalin kieliopin oppiminen

Yleensä oletetaan, että formaali kielioppi on 'annettu' eli esim. asiantuntijan muodostama

Kielioppi voidaan myös oppia ('grammatical inference') opetusnäytteistä

Opetusjoukko H voi koostua positiivisista S^+ ja negatiivisista S^- opetusnäytteistä eli $H = \{S^+, S^-\}$

Tavoitteena on oppia kielioppi G_{learn} siten, että näytteet S^+ kuuluvat kieleen $L(G_{\text{learn}})$, mutta näytteet S^- eivät

Voidaan myös 'ektrapoloida' opetusnäytteitä. Jos

$$S^+ = \{ab, aabb, aaabbb, aaaabbbb\}, \tag{180}$$

voidaan päätellä, että

$$L(G_{\text{learn}}) = \{a^n b^n | n \geq 1\} \text{ ja} \tag{181}$$

$$P = \{S \rightarrow ab, S \rightarrow aSb\} \tag{182}$$

Ongelmallista on se, että sama kieli voi vastata useampaa kuin yhtä kielioppia (mikä kielioppi opitaan?) Tämän takia joudutaan asettamaan opittavalle kieliopille lisärajoituksia (esim. valitaan jokin edellä esitellyistä kielioppityypeistä ja mahdollisimman yksinkertaiset muodostussäännöt)

Opetusjoukon kokoon kannattaa kiinnittää huomiota. Usein $|L(G)| = \infty$ ja käytännössä aina $|S^+| \lll |L(G)|$. Kannattaakin valita S^+ siten, että siinä on käytetty kaikkia G:n muodostussääntöjä

Formaali kielioppi voidaan muodostaa opetusjoukon avulla seuraavasti:

- Annettu: S^+ ja S^-
- Oletettu: $G^{(0)}_{\mathsf{learn}} = \{V_N^{(0)}, V_T^{(0)}, P^{(0)}, S^{(0)}\}$ ja P:n, V_N:n ja V_T:n jäsenien muokkaus- ja luomissäännöt
- Tavoite: löytää $G^{(N)}_{\mathsf{learn}}$ siten, että S^+ kuuluu ja S^- ei kuulu kieleen $L(G^{(N)}_{\mathsf{learn}})$
- Oppimisprosessi:

- aseta $k = 0$
- valitse opetusnäyte $x^{(i)} \in S^+$. Jos sen jäsennys onnistuu, jatka; muuten muokkaa $G^{(k)}_{\mathsf{learn}}$:ta
- valitse opetusnäyte $x^{(i)} \in S^-$. Jos sen jäsennys ei onnistu, jatka; muuten muokkaa $G^{(k)}_{\mathsf{learn}}$:ta
- Jos kaikki opetusnäytteen on käsitelty, lopeta; muuten $k = k + 1$

Edellä esitetyn menetelmän heikkous on se, että P:n, V_N:n ja V_T:n muokkaussääntöjen suunnittelu on hankalaa ja jos säännöt eivät ole yksiselitteiset, mahdollisten kielioppien G^{k+1}_{learn} lkm kasvaa hyvin nopeasti

9.3 Symbolijonojen tunnistus

Edellä tarkasteltiin, kuinka symbolijonoina esitetyt hahmot voidaan mallintaa (generoida) formaalin kieliopin tai vastaavan tilakoneen avulla

Seuraavaksi tarkastellaan kuinka voidaan päätellä onko symbolijono jonkin tietyn kieliopin mukainen tai kuuluuko se johonkin kieleen (tunnistus)

Symbolijonon tunnistus voi perustua joko vertailuun tai jäsentämiseen

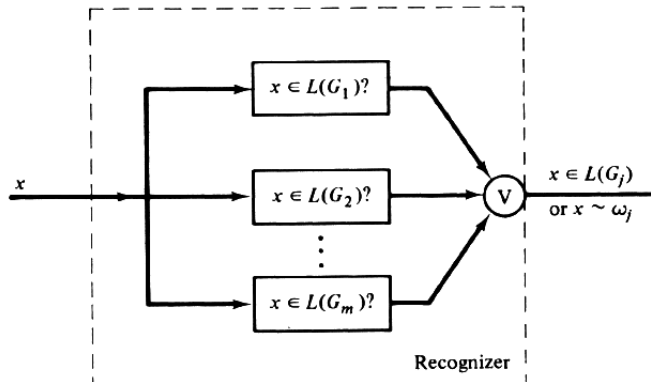
Symbolijonojen jäsennys

Symbolijonon jäsentämisellä tarkoitetaan sitä, että tarkistetaan onko se syntaktisesti hyväksyttävä eli voidaanko se muodostaa kieliopin sääntöjen mukaisesti

Jäsentämisen etuna symbolijonojen vertailuun nähden on se, että samalla saadaan selville symbolijonon (hahmon) sisäinen rakenne

Edellä esitettiin menetelmiä, joiden avulla säännölliset ja kontekstiriippumattomat kieliopit voitiin esittää tilakoneiden ja puiden avulla. Näitä voidaan käyttää symbolijonojen jäsentämiseen

Kaaviokuva syntaktisesta tunnistuksesta:



Syntaktinen tunnistusongelma voidaan siis formuloida seuraavasti:

$$x \in L(G_i) \text{ jollekin } i = 1, \ldots, M? \tag{183}$$

(x on symbolijono (havainto) ja G_i on luokkaa ω_i vastaava kielioppi)

Tarkastellaan tästä eteenpäin kontekstiriippumattomia kieliä

Ol., että G on kontekstiriippumaton kielioppi ja x on jokin symbolijono

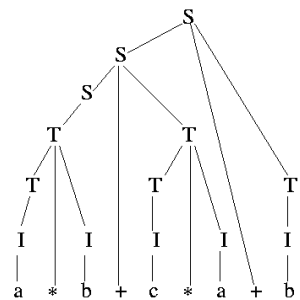
Yritetään muodostaa x:ää vastaavaa jäsennyspuu käyttäen G:n muodostussääntöjä

Mikäli yritys onnistuu, x kuuluu kieleen $L(G)$; muuten ei

Esimerkki: Tarkastellaan kielioppia $G = (V_N, V_T, P, S)$, missä $V_N = \{S, T\}$, $V_T = \{I, +, *\}$, $I \in \{a, b, c\}$ ja

$$P = \{S \to T, \quad T \to T * I, \quad S \to S + T, \quad T \to I\}$$

Kieleen $L(G)$ kuuluva symbolijono $x = a * b + c * a + b$ voidaan jäsentää seuraavasti:



Ei ole merkitystä muodostetaanko puu lähtien ylhäältä aloitussymbolista (juuresta) vai alhaalta terminaalisymboleista (lehdet). Edellistä tapaa kutsutaan 'top-down'- ja jälkimmäistä 'bottom-up'-jäsentämiseksi

Molemmissa tavoissa pyritään yleensä käsittelemään symbolit järjestyksessä vasemmalta oikealle

Jäsennyksessä käytetään yksi kerrallaan muodostussääntöjä, jotka ovat lokaalisti sopivia

Voi käydä niin, että myöhemmin huomataan jonkin muodostussäännön valinnan johtaneen jäsennysyrityksen epäonnistumiseen, vaikka symbolijono kuuluisikin tarkasteltavaan kieleen. Silloin pitää palata taaksepäin ja tehdä erilainen valinta ('backtracking')

Jotta jäsennys olisi tehokasta, voidaan muodostussääntöjen valintaan käyttää *a priori*-sääntöjä, esim. lokaali sopivuus, johtaako lopulta terminaalisymboleihin, joita esiintyy tarkasteltavassa symbolijonossa ja joita on vähemmän kuin vielä käsittelemättömiä symboleita

Jäsennyksen viemä aika ja tehokkuus riippuvat siitä, kuinka hyvin voidaan välttää väärien muodostussääntöjen valinta

Seuraavaksi tarkastellaan lähemmin 'top-down'- ja 'bottom-up'-lähestymistapoja

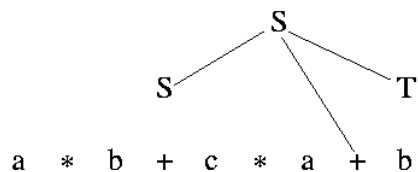
'Top-down'-jäsennys:

- Jäsennys alkaa aloitussymbolista S
- Jäsennys etenee tavoiteorientoituneesti eli tutkitaan voidaanko S hajottaa osatavoitteiksi eli $S \rightarrow X_1 X_2 \ldots X_n$
- Jos X_1 on terminaalisymboli, tarkasteltavan symbolijonon x pitää alkaa tällä
- Jos X_1 on nonterminaalisymboli, otetaan se tarkasteltavaksi osatavoitteeksi ja tutkitaan vastaako x:n alku sitä
- Jos osatavoite X_1 saavutetaan, otetaan X_2 uudeksi tarkasteltavaksi osatavoiteeksi jne

- Mikäli jokin osatavoitteista X_i jää saavuttamatta, kokeillaan uutta aloitusta $S \rightarrow X'_1 X'_2 \ldots X'_n$
- Osatavoitteetkin voidaan hajottaa edelleen uusiksi osatavoitteiksi
- Mikäli jokin osatavoitteista jää saavuttamatta, palataan ylemmälle tasolle ja yritetään uutta osatavoitteiden jakoa
- Huom! vasemmalta oikealle eteneminen ja muotoa $A \rightarrow A\alpha$ olevat rekursiiviset muodostussäännöt voivat yhdessä aiheuttaa ikuisia silmukoita (vältellään näiden sääntöjen käyttöä viimeiseen asti!)
- Tehokas *a priori*-sääntö osatavoitteiden valinnalle: tutkitaan, että vasemmanpuoleisin, käsittelemätön terminaalisymboli voidaan todella johtaa vasemmanpuoleisimmasta, käsittelemättömästä osatavoitteesta (valmiiksi laskettu taulukko!)

- Joitain syitä valita 'top-down'-lähestymistapa:
  - Kielioppi vastaa suoraan hahmojen analysointialgoritmia
  - Takaisin palaamisiin hukattu aika on vähäinen verrattuna terminaalimuuttujien tunnistuksiin käytettyyn aikaan
  - Tavoiteorientoituneisuus on hyödyllistä myös terminaalimuuttujien tunnistukselle

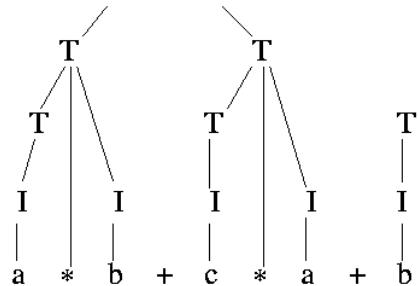
Esimerkki:



'Bottom-up'-jäsennys:

- Jäsennys lähtee liikkeelle tarkasteltavasta symbolijonosta x
- Muodostussääntöjä käytetään 'takaperin'
- Ei osatavoitteita; pyritään vain päätymään lopulta aloitussymboliin
- 'bottom-up'-jäsennys ei ole erityisen tehokas: vääriä valintoja voi olla hyvinkin paljon
- Tehokas *a priori*-sääntö muodostussääntöjen valinnalle: tutkitaan mitkä nonterminaalisymbolit voivat alkaa tietyillä, toisilla nonterminaalisymboleilla (valmiiksi laskettu taulukko!)
- Muotoa $A \rightarrow A\alpha$ olevat rekursiiviset muodostussäännöt eivät ole ongelma
- Mikäli joudutaan umpikujaan, palataan alemmalle tasolle ja kokeillaan vaihtoehtoisia muodostussääntöjä

Esimerkki:



Jäsennys voidaan toteuttaa myös rinnakkaislaskentana: jokaisessa valintatilanteessa monistetaan tarkasteltava ratkaisuyritys, ratkaisuyrityksiä jatketaan rinnakkain, umpikujaan päätyvät yritykset hylätään

'Bottom-up'- ja 'top-down'-lähestymistapojen keskinäinen vertailu on vaikeaa. Toisille kieliopeille 'bottom-up' on tehokkaampi, toisille taas 'top-down'. Lisäksi kieliopin muuttaminen uuteen muotoon vaikuttaa lähestymistapojen tehokkuuksiin

Symbolijonojen vertailu

Yksinkertaisin tapa tunnistaa kieleen kuuluvia symbolijonoja on muodostaa sanakirja: luetellaan (kaikki) kieleen kuuluvat symbolijonot ja verrataan havaittua symbolijonoa näihin

Tunnistus voi perustua esim. k:n lähimmän naapurin menetelmään

Lähestymistavan heikkous on se, että usein $|L(G)| = \infty$

Symbolijonojen vertailuun tarvitaan jokin mitta, joka kuvaa niiden samankaltaisuutta

Yleensä esim. Euklidinen metriikka ei käy samankaltaisuuden mitaksi (vertailtavat symbolijonot voivat olla eri pituisia!)

Samankaltaisuusmitan tulee huomioida sekä symbolijonojen symboleiden samankaltaisuus että niiden sisäinen rakenne (esim. sanojen oikeinkirjoitus ja ääntäminen (ruotsi-ruatsi ja betoni-petonni) tai taivutus)

Samankaltaisuusmitan valinta riippuu luonnollisesti hyvin paljon hahmojen esitystavan (symbolit) valinnasta ja sovellutusongelmasta ominaispiirteistä (rakenteiden merkitys)

Esimerkki: käsinkirjoitetut merkit voidaan esittää symbolijonoina, joissa eri symbolit vastaavat vakiopituisia, mutta eri suuntaisia kaaren pätkiä (*'chain coding'*). Symboleiden vaihtuminen toisiksi voi merkitä, että jokin merkin osa on piirretty vähän eri suuntaan. Symboleiden puuttuminen tai lisääntyminen voi tarkoittaa, että jokin merkin osa on pienempi tai suurempi

Tarkastellaan aluksi yleisempää ongelmaa eli aikasarjojen vertailua

Jos aikasarjan datapisteiden (piirrevektoreiden) välille on määritelty sovituskustannus, esim. Euklidinen etäisyys, voidaan käyttää **Dynamic Time Warping-algoritmia** (DTW):

- DTW-algoritmi sovittaa kahden aikasarjan datapisteet siten, että sovitettujen datapisteparien kustannuksien summa (tai tulo) on mahdollisimman pieni

- Olkoot $\mathbf{r}(i),\ i = 1, \ldots, I$, ja $\mathbf{t}(j),\ j = 1, \ldots, J$, kaksi aikasarjaa. Yleensä $I \neq J$
- Määritellään 2-ulotteinen avaruus, jonka piste (i, j) vastaa aikasarjojen datapisteitä $\mathbf{r}(i)$ ja $\mathbf{t}(j)$:
- Aikasarjojen sovitus voidaan esittää seuraavan polun avulla:

$$(i_0, j_0), (i_1, j_1), \ldots, (i_f, j_f) \tag{184}$$

- Jokaista polkua vastaa kokonaiskustannus D

$$D = \sum_{k=0}^{K-1} d(i_k, j_k), \text{ tai} \tag{185}$$

$$D = \sum_{k=0}^{K-1} d(i_k, j_k | i_{k-1}, j_{k-1}), \text{ tai} \tag{186}$$

$$D = \prod_{k=0}^{K-1} d(i_k, j_k | i_{k-1}, j_{k-1}), \tag{187}$$

missä K on sovitettujen pisteparien lkm (polun pituus), ja $d(i_k, j_k)$ ja $d(i_k, j_k|i_{k-1}, j_{k-1})$ ovat datapisteiden $\mathbf{r}(i_k)$:n ja $\mathbf{t}(j_k)$:n väliset sovituskustannukset silloin, kun edelliset sovitukset joko ei huomioida tai huomioidaan

- Poluille voidaan asettaa erilaisia rajoituksia, esim. kuinka sovitetaan aikasarjojen ensimmäiset ja viimeiset pisteet, kuinka paljon polku voi poiketa lineaarisesta sovituksesta lokaalisti ja globaalisti jne

- DTW-algoritmin ratkaisu perustuu *Bellmannin optimaalisuus-periaatteeseen:*

  - Merkitään optimaalista polkua seuraavasti:

$$(i_0, j_0) \rightarrow^{\mathsf{opt}} (i_f, j_f) \tag{188}$$

  - Mikäli optimaalinen polku kulkee pisteen (i, j) kautta, voidaan merkitä:

$$(i_0, j_0) \rightarrow^{\mathsf{opt}}_{(i,j)} (i_f, j_f) \tag{189}$$

– Bellmannin periaatteen mukaan

$$(i_0, j_0) \rightarrow^{\mathsf{opt}}_{(i,j)} (i_f, j_f) = (i_0, j_0) \rightarrow^{\mathsf{opt}} (i, j) \oplus (i, j) \rightarrow^{\mathsf{opt}} (i_f, j_f), \tag{190}$$

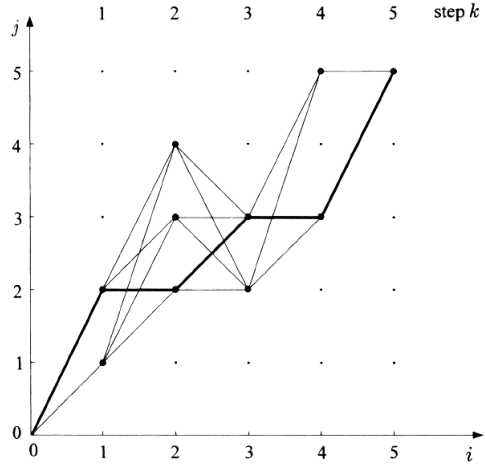
missä $\oplus$ tarkoittaa polkujen yhdistämistä

– Edellisen perusteella optimaalinen polku voidaan ratkaista rekursiivisesti:

$$D_{\mathsf{min}}(i_k, j_k) = \min_{(i_{k-1}, j_{k-1})} [D_{\mathsf{min}}(i_{k-1}, j_{k-1}) + d(i_k, j_k | i_{k-1}, j_{k-1})] \tag{191}$$

(jos polkua vastaava kokonaiskustannus on tulomuotoinen korvataan summa tulolla)

– Esimerkki rekursiivisesta ratkaisun etsimisestä:



- DTW-etäisyyttä on käytetty paljon esim. puheen ja käsinkirjoitetun tekstin tunnistuksessa

Yksi paljon käytetty DTW-algoritmiin perustuva mitta symbolijonojen vertailussa on **editointietäisyys:**

- Oletetaan, että hahmot koostuvat symbolijonoista (ja symbolien järjestyksellä on väliä!)
- Mitataan kahden symbolijonon, $\mathbf{A}(i),\ i = 1, \ldots, I$, ja $\mathbf{B}(j),\ j = 1, \ldots, J$, samankaltaisuutta tarvittavien editointioperaatioiden lkm:ien (symbolin vaihtoja C, lisäyksiä I ja poistoja R) avulla:

  $$D(\mathbf{A}, \mathbf{B}) = \min_j [C(j) + I(j) + R(j)], \tag{192}$$

  missä j indeksoi kaikki mahdolliset muokkaukset, joilla $\mathbf{B}$:stä saadaan $\mathbf{A}$
- Määritellään 2-ulotteinen avaruus, jonka piste (i, j) vastaa symboleita $\mathbf{A}(i)$ ja $\mathbf{B}(j)$
- Etsitään optimaalista sovituspolkua pisteestä $(0, 0)$ pisteeseen (I, J)

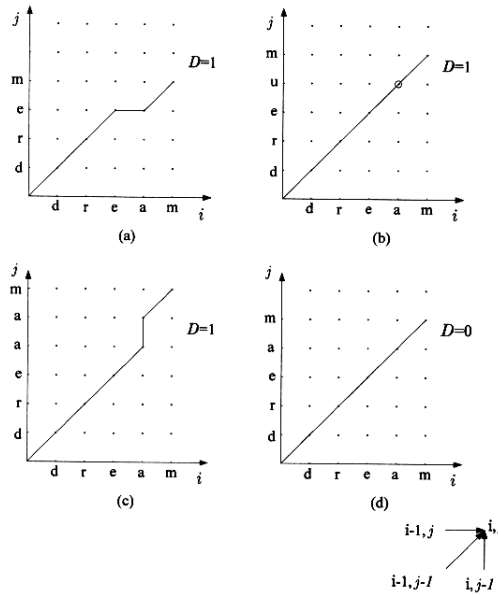
- Määritellään polun pisteelle $(0,0)$ kustannus $D(0,0) = 0$
- Polun pisteeseen (i,j) voidaan päästä vain pisteistä $(i-1,j)$, $(i-1,j-1)$ tai $(i,j-1)$
- Symbolien sovituksiin liitetään seuraavat kustannukset:

$$d(i,j|i-1,j-1) = \begin{cases} 0, \text{ jos } \mathbf{A}(i) = \mathbf{B}(j) \\ 1, \text{ jos } \mathbf{A}(i) \neq \mathbf{B}(j) \text{ (symbolin vaihto)} \end{cases} \tag{193}$$

$$d(i,j|i-1,j) = d(i,j|i,j-1) = 1 \text{ (symbolin poisto tai lisäys)} \tag{194}$$

- Optimaalinen sovituspolku voidaan nyt ratkaista rekursiivisesti

- Esimerkkejä:



Symbolijonojen vertailuun perustuva tunnistus on järkevää ainoastaan silloin, kun voidaan muodostaa ongelmaan hyvin sopiva sanakirja ja samankaltaisuuden mitta

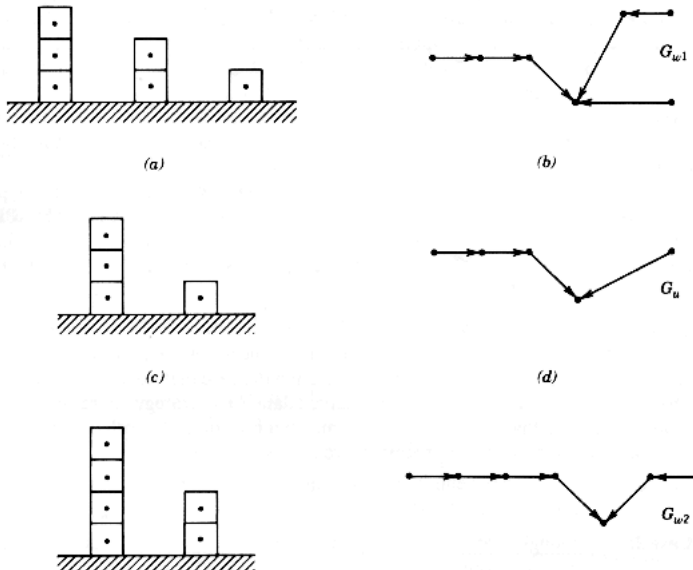
(symbolijonojen vertailun eräänlaisena laajennuksena voidaan pitää graafimuotoisina esitettyjen hahmojen vertailua)

9.4 Graafien vertailuun perustuva tunnistus

Suunnatun graafin avulla voidaan esittää monimutkaisempia piirteiden välisiä suhteita kuin yksiulotteisilla symbolijonoilla

Ajatellaan, että graafin $G = \{N, R\}$ solmut N ovat (erilaisia) piirteitä ja niitä yhdistävät kaaret R kuvastavat (erilaisia) piirteiden välisiä suhteita (atribuuttigraafi)

Esimerkki graafeina esitetyistä hahmoista:



Tunnistus perustuu havaitun hahmon graafin ja eri luokkien prototyyppigraafien vertailuun ja esim. k:n lähimmän naapurin menetelmään

Havainnon graafi ei välttämättä vastaa täydellisesti mitään prototyyppigraafia. Havainnon graafi voi olla esim. vain jokin prototyypin osagraafi

Graafien välinen samankaltaisuusmitta perustuu sekä graafien solmujen että niitä yhdistävien kaarien samankaltaisuuteen

Graafit voidaan esittää $p \times p$-kokoisen vierekkäisyysmatriisin M ('adjacency matrix') avulla, missä p on solmujen lkm ja $M(i,j) \neq 0$, jos solmujen i ja j välillä on jonkinlainen kaari. Muuten $M(i,j) = 0$

Graafien homo- ja isomorfismi

Graafien $G_1 = \{N_1, R_1\}$ ja $G_2 = \{N_2, R_2\}$ sanotaan olevan *homomorfisia*, jos on olemassa kuvaus $f: N_1 \rightarrow N_2$ siten, että jos solmujen $v_1, w_1 \in N_1$ välillä on kaari graafissa G_1, niin silloin myös graafissa G_2 on (samanlainen)

kaari solmujen $f(v_1), f(w_1) \in N_2$ välillä eli

$$(v_1, w_1) \in R_1 \Rightarrow (f(v_1), f(w_1)) \in R_2 \tag{195}$$

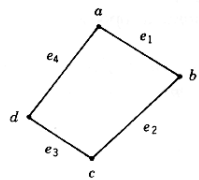
Graafit ovat *isomorfisia*, jos ne ovat homomorfisia ja kuvaus f on 1:1 ja kaikki N_2:n alkiot ovat jonkin N_1:n alkion kuvapisteitä (bijektio) eli

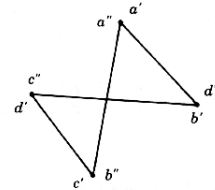
$$(v_1, w_1) \in R_1 \Leftrightarrow (f(v_1), f(w_1)) \in R_2 \tag{196}$$

(toisin sanoen graafista G_1 saadaan graafi G_2 indeksoimalla solmut uudestaan)

Subisomorifismi tarkoittaa sitä, että graafin G_1 osagraafi on isomorfinen jonkin graafin G_2 osagraafin kanssa

Esimerkki isomorfisista graafeista:





(Symmetric adj. matrix)

	a	b	c	d
a	0	1	0	1
b	1	0	1	0
c	0	1	0	1
d	1	0	1	0

	a'	b'	c'	d'
a'	0	1	1	0
b'	1	0	0	1
c'	1	0	0	1
d'	0	1	1	0

$f(a) = a''$
$f(b) = b''$
$f(c) = c''$
$f(d) = d''$

	a''	b''	c''	d''
a''	0	1	0	1
b''	1	0	1	0
c''	0	1	0	1
d''	1	0	1	0

Käytännönongelmissa graafien isomorfisuus ei ole hyvä luokittelukriteeri, koska se ei salli graafeille pieniäkään poikkeamia eikä siksi siedä lainkaan virheitä esim. piirreirrotuksessa

Graafien isomorfisuuden toteaminen on lisäksi laskennallisesti hyvin raskasta:

- Vertaillaan graafeja G_1 ja G_2, jotka on esitetty vierekkäisyysmatriisien M_1 ja M_2 avulla
- Jos $M_1 = M_2$, graafit ovat isomorfisia. Tarkistus joudutaan tekemään $p \times p$-matriisin kaikille alkioille ($O(|N_1|^2)$)
- Jos $M_1 \neq M_2$, voidaan kokeilla kaikkia muita (yht. $p!$) tapoja indeksoida graafin M_1 solmut (vaihtoehtojen etsintä $O(p^k)$, missä k on vaihtoehtojen lkm)

Usein käytetäänkin testejä, jotka epäonnistuvat, jos graafit eivät ole isomorfisia. Seuraavat graafien ominaisuudet ovat invariantteja isomorfismin suhteen:

- (tietynlaisten) solmujen lkm
- (tietynlaisten) kaarien lkm
- (tietynlaisten) solmuun tulevien, solmusta lähtevien tai solmuun liittyvien, suuntaamattomien kaarien lkm:t
- l:n pituiset syklit

Graafien vertailuun on kehitetty useita erilaisia menetelmiä, jotka sallivat poikkeamat graafien välillä. Kaksi tyypillistä lähestymistapaa:

- Muodostetaan graafeille piirrevektorit ja vertaillaan niitä

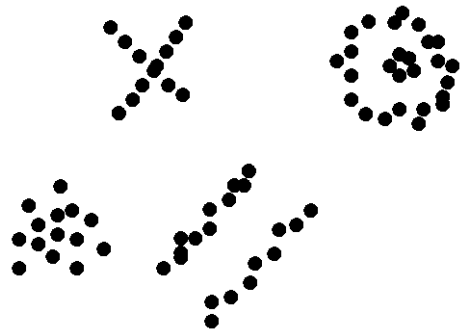
- Käytetään graafien samankaltaisuuden mittana pienintä tarvittavien editointioperaatioden lkm:ää, joilla graafit saadaan muutettua toisikseen:
  - solmujen lisäys ja poisto
  - solmujen yhdistäminen ja jakaminen
  - solmujen tyypin vaihto
  - kaarien lisäys ja poisto
  - kaaren tyypin vaihto

Edellisten lähestymistapojen ongelmana on sopivien piirteiden ja metriikan valinta tai laskennallinen kompleksisuus

10. OHJAAMATON OPPIMINEN JA KLUSTEROINTI

Ohjaamattomassa oppimisessa on tavoitteena muodostaa hahmoista ryhmiä, klustereita, joiden sisällä hahmot ovat jossain mielessä samankaltaisia ja joiden välillä on selkeitä eroja

Erilaisia klustereita:



Käytettävissä oleville havainnoille ei ole valmiina mitään luokitustietoa ('labeling'). Myös klustereiden lkm voi olla tuntematon

Ohjaamattoman oppimisen menetelmiä voidaan myös käyttää ongelmissa, joissa havainnoilla on luokitustieto, mutta halutaan selvittää luokkien sisäinen rakenne

Ohjaamattoman oppimisen menetelmät voidaan jakaa lähestymistavoiltaan kahteen luokkaan:

- Parametriset menetelmät: yritetään yhtäaikasesti selvittää sekä havaintojen luokittelu että luokkien tnjakaumien parametrit (mikstuurimallit & EM-algoritmi!)
- Epäparametriset menetelmät: jaetaan havainnot eri luokkia vastaaviksi osajoukoiksi, luokkien tnjakaumia ei estimoida

Ensimmäinen lähestymistapa on suoraviivainen laajennus tilastolliselle hahmontunnistukselle; jälkimmänen lähestymistapa on tilastollisessa mielessä epäoptimaalinen, mutta laskennallisesti yleensä huomattavasti helpommin käsiteltävä

Klusterointiongelman ratkaiseminen voidaan jakaa seuraaviin vaiheisiin:

- Piirteiden valinta (esikäsittely & normalisointi!)
- Samankaltaisuus/erilaisuusmitan valinta havaintoparien, havaintojen ja klustereiden, sekä klusteriparien välille (piirteiden samanarvoinen kohtelu!)
- Klusterikriteerin valinta
- Klusterointialgoritmin valinta
- Tulosten validiointi erilaisten testien avulla
- Tulosten tulkinta

Huom! Saatu ratkaisu on erittäin subjektiivinen, koska sama havaintojoukko voidaan jakaa hyvin erilaisiin klustereihin riippuen em valinnoista. Yleensä ratkaisun tulkinnan ja sen hyvyyden arvioinnin suorittaa sovellusalan asiantuntija

Havaintojen klusterointia voidaan käyttää luokittelun lisäksi seuraaviin tarkoituksiin:

- Datan tiivistys: havainto esitetään piirrevektorin sijasta klusterin koodinumeron avulla (LVQ!)
- Havaintoihin liittyvien hypoteesien muodostus (SOM!) ja testaus
- Havaintoihin perustuva ennustaminen: havainnon puuttuvat tiedot ennustetaan muiden samaan klusteriin kuuluvien havaintojen perusteella

- Luonnolliset klusterit ('natural clusters'): piirreavaruuden alueita, joissa havainnot ovat suhteellisen tiheässä ja joiden välissä havainnot ovat suhteellisen harvassa
- Todennäköisyyksiin perustuvat klusterit: havainto $\mathbf{x}$ kuuluu siihen klusteriin C_k, jonka *a posteriori* tn on korkein eli $P(C_k|\mathbf{x}) \geq P(C_i|\mathbf{x})\ \forall i = 1, \ldots, m$
- Yksikäsitteinen ('hard', 'crisp') klusterointi:

$$C_i \neq \emptyset,\ i = 1, \ldots, m \tag{197}$$
$$\cup_{i=1}^{m} C_i = X \tag{198}$$
$$C_i \cap C_j = \emptyset,\ i \neq j,\ i, j = 1, \ldots, m \tag{199}$$

(C_i on i. klusteri, m klustereiden lkm ja X kaikki havainnot)

- Sumea ('fuzzy') klusterointi. Havainnon kuulumisasteet eri klusterihin ilmaistaan jäsenyysfunktioiden ('membership functions') avulla:

$$u_j : X \to [0,1],\ j = 1, \ldots, m \tag{200}$$

$$\sum_{j=1}^{m} u_j(\mathbf{x}_i) = 1,\ i = 1, \ldots, N \tag{201}$$

$$0 < \sum_{i=1}^{N} u_j(\mathbf{x}_i) < N,\ j = 1, \ldots, m \tag{202}$$

($\mathbf{x}_i$ on i. havainto, u_j klusteriin C_j liittyvä kuuluvuusfunktio, m klustereiden lkm, X kaikki havainnot ja N havaintojen lkm)

10.1 'Sequental clustering algorithms'

Havainnot $\mathbf{x}_1, \ldots, \mathbf{x}_N$ käydään läpi yksitellen, mahdollisesti useammin kuin yhden kerran

Tuottavat havainnoille yhden klusteroinnin

Klustereiden lkm:ää m ei tarvitse tietää etukäteen

Havainnon ja klusterin samankaltaisuutta kuvataan funktion $d(\mathbf{x}, C)$ avulla

Klusteriparin samankaltaisuutta kuvataan funktion $d(C_i, C_j)$ avulla

'Basic Sequental Algorithmic Scheme', BSAS

Klusterointialgoritmin, jossa havainnot käsitellään yksitellen, perusversio:

- Määrätään klustereiden lkm:n yläraja q ja kynnysarvo Θ havainnon ja klusterin samankaltaisuudelle
- Lähdetään liikkeelle yhdestä klusterista: $m = 1$, $C_m = \{\mathbf{x}_1\}$
- Käydään havainnot $\mathbf{x}_i$ läpi yksitellen $i = 2, \ldots, N$ ja
  - Etsitään klusteri C_k, jolle pätee $d(\mathbf{x}_i, C_k) = \min_{1 \leq j \leq m} d(\mathbf{x}_i, C_j)$
  - Jos $d(\mathbf{x}_i, C_k) > \Theta$ ja $m < q$, silloin $m = m + 1$ ja $C_m = \{\mathbf{x}_i\}$
  - Muuten, $C_k = C_k \cup \{\mathbf{x}_i\}$ ja päivitetään $d(\mathbf{x}, C_k)$

Havaintojen läpikäyntijärjestyksellä, funktion $d(\mathbf{x}, C)$ ja kynnysparametrin Θ valinnalla on huomattava vaikutus BSAS-tyyppisen klusterointialgoritmin löytämään ratkaisuun

Funktion $d(\mathbf{x}, C)$ valinta vaikuttaa löydettävien klustereiden rakenteeseen. Jos esim. klusteri C_k esitetään yhden vektorin $\mathbf{m}_k$ avulla ja määritellään $d(\mathbf{x}, C_k) = d(\mathbf{x}, \mathbf{m}_k)$, algoritmilla on taipumus löytää kompakteja klustereita

Kynnysparametri Θ vaikuttaa suoraan löydettävien klustereiden lkm:ään

Sopiva Θ:n arvo (tai klustereiden lkm) voidaan estimoida seuraavasti:

- Kokeillaan erilaisia kynnysparametrin arvoja $\Theta = a, a+c, a+2c, \ldots, b$
- Suoritetaan klusterointi jokaisella kynnysarvolla N kertaa ja käydään havainnot läpi aina erilaisessa satunnaisessa järjestyksessä
- Lasketaan jokaiselle kynnysarvolle keskimääräinen klusterien lkm ja piirretään tuloksista kuvaaja $\bar{m}(\Theta)$
- Kohdat, joissa $\bar{m}(\Theta)$ on pitkään vakioarvoinen, ovat hyviä Θ:n arvoja

BSAS-tyyppisen klusterointialgoritmin varjopuolia:

- Päätös siitä, mihin klusteriin havainto $\mathbf{x}_i$ kuuluu, perustuu vain aikaisemmin käsiteltyihin havaintoihin $\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}$ eikä koko havaintoaineistoon X
- Kerran tehtyä päätöstä uudesta klusterista ei voida perua. Siksi lopullisessa klusteroinnissa voi olla joitain hyvin samankaltaisia klustereita, jotka olisi järkevä yhdistää
- Lopputulos riippuu siitä, missä järjestyksessä havainnot käydään läpi

BSAS-tyyppisille klusterialgoritmille onkin olemassa erilaisia variaatiota, jotka huomioivat paremmin em ongelmat:

- Muodostetaan havaintojen 1.:llä läpikäyntikierroksella m kpl:tta yhdestä havainnosta koostuvia klustereita. Jaetaan seuraavilla kierroksilla loput havainnot näihin klustereihin
- Määrätään kynnysarvo klusteriparin väliselle samankaltaisuudelle ja tutkitaan onko ratkaisussa kaksi klusteria, jotka ovat keskenään liian samankaltaiset. Jos on, yhdistetään samankaltaisin pari ja päivitetään klustereiden numerointi ja samankaltaisuusmitat $d(C_i, C_j)$. Toistetaan, kunnes kaikki klusterit ovat riittävän erilaisia
- Tarkistetaan lopuksi, mitkä ovat eri havaintoja parhaiten vastaavat klusterit ja jaetaan havainnot uudestaan näihin parhaisiin klustereihin. Päivitetään havaintojen ja klustereiden samankaltaisuusmitat $d(\mathbf{x}, C)$. Toistetaan, kunnes havaintojen jako klustereihin ei enää muutu

10.2 Hierarkiset klusterointialgoritmit

Hierarkiset klusterointialgoritmit muodostavat havainnoille klusterointiratkaisujen sarjan, joilla on selkeä sisäkkäinen ('nested') rakenne

Algoritmit voidaan jakaa kahteen alaluokkaan: kasaaviin ('agglomerative') ja pilkkoviin ('divisive') algoritmeihin

Kasaavat algoritmit muodostavat klusterointiratkaisujen sarjan yhdistelemällä klustereita. Algoritmi lähtee liikkeelle tilanteesta, jossa jokainen havainto vastaa yhtä klusteria ($m = N$). Algoritmin edetessä klustereiden lkm pienenee, kunnes kaikki havainnot kuuluvat yhteen klusteriin ($m = 1$)

Pilkkovat algoritmit toimivat päinvastoin. Aluksi kaikki havainnot kuuluvat yhteen klusteriin ($m = 1$) ja algoritmin edetessä klustereita jaetaan kahtia, kunnes jokaisessa klusterissa on tasan yksi havainto ($m = N$)

Kasaava klusterointialgoritmi muodostaa klusterointiratkaisujen sarjan $R_0, \ldots, R_{N-1}$, joilla on seuraava sisäkkäinen rakenne:

$$R_0 \sqsubset R_1 \sqsubset \cdots \sqsubset R_{N-1} \tag{203}$$

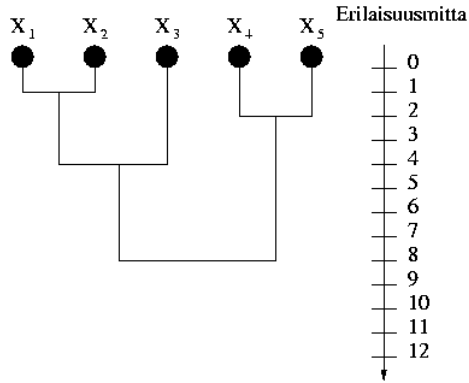
($R_0 = \{\{\mathbf{x}_1\}, \ldots, \{\mathbf{x}_N\}\}$, $R_{N-1} = \{X\} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$)

Pilkkova klusterointialgoritmi muodostaa ratkaisujen sarjan $R_0, \ldots, R_{N-1}$, joilla on seuraava sisäkkäinen rakenne:

$$R_{N-1} \sqsubset R_{N-2} \sqsubset \cdots \sqsubset R_0 \tag{204}$$

($R_0 = \{X\} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $R_{N-1} = \{\{\mathbf{x}_1\}, \ldots, \{\mathbf{x}_N\}\}$)

Hierarkisen klusterointialgoritmin tuottama ratkaisusarja voidaan esittää dendogrammin avulla:



Dendogrammin tasot vastaavat ratkaisuja, joissa on tietty määrä klustereita

Sopivan klustereiden lkm valitsee usein asiantuntija dendogrammin avulla (pitkäikäiset klusterit!)

'Generalized Agglomerative Scheme' GAS

Kasaavan, hierarkisen klusterointialgoritmin yleistetty muoto:

- Klustereiden samankaltaisuutta/erilaisuutta mitataan funktion $g(C_i, C_j)$ avulla
- Alustus: aseta $R_0 = \{C_i = \{\mathbf{x}_i\},\ i = 1, \ldots, N\}$ ja $t = 0$
- Toista kunnes kaikki havainnot ovat yhdessä klusterissa eli $R_t = \{X\}$
  - t=t+1
  - Etsi ratkaisuun R_{t-1} kuuluvista klustereista pari $(C_i.C_j)$, jolle

$$g(C_i, C_j) = \begin{cases} \min_{r,s} g(C_r, C_s), & \text{jos } g \text{ on erilaisuusmitta} \\ \max_{r,s} g(C_r, C_s), & \text{jos } g \text{ on samankaltaisuusmitta} \end{cases} \tag{205}$$

  - Määritä uusi klusteri $C_q = C_i \cup C_j$ ja ratkaisu $R_t = (R_{t-1} - \{C_i, C_j\}) \cup \{C_q\}$

– Päivitä klustereiden numerointi ja funktio g

Edellä esitetyn GAS-tyyppisen klusterointialgoritmin heikkous on se, että klustereiden yhdistämistä ei voida koskaan perua

Askeleella t klustereiden lkm on $N-t$ ja yhdistettävän klusteriparin löytämiseksi on tehtävä

$$\binom{N-t}{2} = \frac{(N-1)(N-t-1)}{2} \tag{206}$$

klustereiden välistä vertailua

Kaikilla askelilla vertailuja tarvitaan yhteensä

$$\sum_{t=0}^{N-1} \binom{N-t}{2} = \sum_{k=1}^{N} \binom{k}{2} = \frac{(N-1)N(N+1)}{6} \tag{207}$$

'Matrix Updating Algorithmic Scheme' MUAS

Eräs GAS-tyyppisten klusterointialgoritmien alalaji on matriisiteoriaan perustuvat MUAS-tyyppiset algoritmit

Tarpeellisia määritelmiä:

- Hahmomatriisi $D(X) = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^T$
- Samankaltaisuus/erilaisuusmatriisi $P(X)$, jonka alkio (i, j) on hahmojen $\mathbf{x}_i$ ja $\mathbf{x}_j$ samankaltaisuutta/erilaisuutta kuvaava funktio $d(\mathbf{x}_i, \mathbf{x}_j)$

MUAS-algoritmin perusversio:

- Alustus: $R_0 = \{\{\mathbf{x}_i\}, \ldots, \{\mathbf{x}_N\}\}$, $P_0 = P(X)$, $t = 0$
- Toista kunnes kaikki havainnot ovat yhdessä klusterissa eli $R_t = \{X\}$
  - t=t+1
  - Etsi klusterit C_i ja C_j s.e. $d(C_i, C_j) = \min_{r,s=1,\ldots,N, r \neq s} d(C_r, C_s)$
  - Määritä uusi klusteri $C_q = C_i \cup C_j$ ja ratkaisu $R_t = (R_{t-1} - \{C_i, C_j\}) \cup \{C_q\}$

– Muodosta matriisi P_t matriisista P_{t-1}

Usein klustereiden erilaisuutta kuvataan funktion $d(C_i, C_j)$ avulla, jonka päivityssääntö voidaan kirjoittaa seuraavassa muodossa:

$$\begin{aligned} d(C_q, C_s) =& a_i d(C_i, C_s) + a_j d(C_j, C_s) + b d(C_i, C_j) \\ &+ c|d(C_i, C_s) - d(C_j, C_s)| \end{aligned} \tag{208}$$

Parametrien a_i, a_j, b ja c arvot riippuvat valitusta funktiosta $d(C_i, C_j)$

Kun $a_i = a_j = 1/2$, $b = 0$ ja $c = -1/2$ eli

$$d(C_q, C_s) = \min\{d(C_i, C_s), d(C_j, C_s)\}, \tag{209}$$

kyseessä on *'single link'* klusterointialgoritmi

Kun $a_i = a_j = 1/2$, $b = 0$ ja $c = 1/2$ eli

$$d(C_q, C_s) = \max\{d(C_i, C_s), d(C_j, C_s)\}, \tag{210}$$

kyseessä on *'complete link'* klusterointialgoritmi

'Single link' suosii pitkulaisia klustereita; 'complete link' taas kompakteja klustereita

'Graph Theory-based Algorithmic Scheme' GTAS

GTAS-tyyppiset algoritmit ovat GAS-tyyppisten algoritmien alalaji, jotka perustuvat graafiteoriaan

Näissä algoritmeissä havaintojoukko X esitetään graafin avulla s.e. jokainen solmu vastaa yhtä havaintoa ja solmut, joiden välillä on polku, kuuluvat samaan klusteriin

Solmujen yhtenäisyyden lisäksi voidaan klustereille asettaa lisävaatimus $h(k)$, esim:

- Kaikki klusterin solmuparit on kytketty toisiinsa vähintään k:lla polulla, joilla ei ole yhteisiä solmuja
- Kaikki klusterin solmuparit on kytketty toisiinsa vähintään k:lla polulla, joilla ei ole yhteisiä kaaria
- Jokaisen solmun asteluku on vähintään k

GTAS-tyyppinen algoritmi saadaan suoraan GAS-algoritmin perusversiosta korvaamalla $g(C_i, C_j)$ rajoitusehdon $h(k)$ huomioivalla funktiolla $g_{h(k)}(C_i, C_j)$

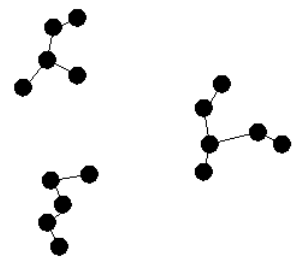
MST-algoritmi on eräs GTAS-tyyppinen algoritmi, joka perustuu minivirittäjäpuuhun ('Minimum Spanning Tree') ja jolle

$$g(C_r, C_s) = \min_{i,j}\{w_{ij} : \mathbf{x}_i \in C_r, \mathbf{x}_j \in C_s\}, \tag{211}$$

missä $w_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$

MST-algoritmi ja 'single link' tuottavat saman ratkaisun, mikäli $d(\mathbf{x}_i, \mathbf{x}_j) \neq d(\mathbf{x}_k, \mathbf{x}_l)$, kun $i \neq k$ tai $j \neq l$

Esimerkki MST-algoritmilla muodostetusta klustroinnista:



'Generalized Divisive Scheme' GDS

Pilkkovan klusterointialgoritmin yleistetty muoto:

- Klustereiden erilaisuutta mitataan funktion $g(C_i, C_j)$ avulla
- Alustus: aseta $R_0 = \{X\}$, $t = 0$
- Toista kunnes jokainen havainto vastaa yhtä klusteria eli $R_t = \{\{\mathbf{x}_1\}, \ldots \{\mathbf{x}_N\}\}$
  - $t = t + 1$
  - Käy läpi kaikki klusterit $C_i,\ i = 1, \ldots, t$, ja etsi jaot (C_i^1, C_i^2), jotka maksimoivat $g(C_i^1, C_i^2)$:n
  - Pilko klusteri $C_j = \arg\max_{C_i, i=1,\ldots,t} g(C_i^1, C_i^2)$
  - Määritä uusi ratkaisu $R_t = (R_{t-1} - \{C_j\}) \cup \{C_j^1, C_j^2\}$
  - Päivitä klustereiden numerointi ja funktio g

GDS-tyyppinen klusterointialgoritmi on laskennallisesti hyvin vaativa

10.3 Funktion optimointiin perustuvat klusterointialgoritmit

Klusteroinnin onnistumista mittaavan funktion J optimointiin perustuvissa klusterointialgoritmeissä ratkaisu esitetään usein parametrivektorin $\mathbf{\Theta} = (\mathbf{\Theta}_1^T, \ldots, \mathbf{\Theta}_m^T)^T$, missä $\mathbf{\Theta}_i$ on klusteriin C_i liittyvät parametrit, avulla

Parametrivektoreiden avulla voidaan määrittää löydettävien klustereiden rakenne (esim. hypertaso, hyperpallon kuori, toruksen kuori jne). Tämä on hyödyllinen ominaisuus esim. tietokonenäön sovellutuksista (kohteen ja taustan erottaminen toisistaan)

Tehdään siis *a priori* oletus klustereiden rakenteesta ja esitetään se parametrien avulla

Klustereden lkm m oletetaan yleensä tunnetuksi

'C-means'-algoritmin variaatio

Klusterit $C_1, \ldots, C_c$ esitetään prototyyppivektoreiden $\mathbf{\Theta} = (\mathbf{m}_1^T, \ldots, \mathbf{m}_c^T)^T$ avulla ja klusteroinnin onnistumisen mittaamiseen käytetään seuraavanlaista kustannusfunktiota:

$$J_{SSE} = \sum_{i=1}^{c} \sum_{\mathbf{x}_k \in C_i} \|\mathbf{x}_k - \mathbf{m}_i\|^2 \tag{212}$$

J_{SSE}:n minimi voidaan löytää seuraavalla algoritmilla:

- Valitaan prototyyppivektoreille $\mathbf{m}_1, \ldots, \mathbf{m}_c$ satunnaiset alkuarvot
- Toista kunnes klusterointi eli prototyyppivektorit eivät enää muutu
  - Jaetaan havainnot klustereihin s.e. $\mathbf{x} \in C_j$, jos

$$\frac{N_j}{N_j + 1} \|\mathbf{x} - \mathbf{m}_j\|^2 \leq \frac{N_i}{N_i - 1} \|\mathbf{x} - \mathbf{m}_i\|^2 \; \forall i = 1, \ldots, c, \tag{213}$$

missä N_i on klusteriin C_i kuuluvien havaintojen lkm

– Päivitetään prototyyppivektorit seuraavasti: $\mathbf{m}_k = \frac{1}{N_k} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$

Todistus:

- Oletetaan, että havainto $\mathbf{x}_j$ siirretään klusterista C_i klusteriin C_j
- Uuden klusterin C'_j prototyyppivektori on

$$\mathbf{m}'_j = \frac{1}{N_j + 1} \sum_{\mathbf{x}_k \in C'_j} \mathbf{x}_k = \mathbf{m}_j + \frac{\mathbf{x}_j - \mathbf{m}_j}{N_j + 1} \tag{214}$$

- Uuteen klusteriin C'_j liittyvä kustannus on silloin

$$J_{SSE}(C'_j) = \sum_{\mathbf{x}_k \in C'_j} \|\mathbf{x}_k - \mathbf{m}'_j\|^2 = J_{SSE}(C_j) + \frac{N_j}{N_j + 1} \|\mathbf{x}_j - \mathbf{m}_j\|^2 \tag{215}$$

- Vastaavasti, uuteen klusteriin C_i' liittyvä kustannus on

$$J_{SSE}(C_i') = J_{SSE}(C_i) - \frac{N_i}{N_i - 1}\|\mathbf{x}_j - \mathbf{m}_i\|^2 \tag{216}$$

- J_{SSE}:n kokonaismuutos on

$$\Delta J_{SSE} = \frac{N_j}{N_j + 1}\|\mathbf{x}_j - \mathbf{m}_k\|^2 - \frac{N_i}{N_i - 1}\|\mathbf{x} - \mathbf{m}_i\|^2 \tag{217}$$

('C-means'-algoritmin perusversio ei ota kantaa kuinka havainnot jaetaan klustereihin ja kuinka klustereiden prototyyppivektoreita päivitetään)

10.4 Kilpailuun perustuvat klusterointialgoritmit

Kilpailuun perustuvissa klusterointialgoritmeissä klusterit $C_1, \ldots, C_m$ esitetään yleensä vektoreiden $\mathbf{w}_1, \ldots, \mathbf{w}_m$ avulla

Vektoreita (klustereita) $\mathbf{w}_1, \ldots, \mathbf{w}_m$ päivitetään havaintojen perusteella s.e. ne siirtyvät niihin kohtiin avaruutta, joissa havaintoja on tiheässä

Vektorit $\mathbf{w}_1, \ldots, \mathbf{w}_m$ kilpailevat keskenään. Havaintoa parhaiten vastaavaa voittajavektoria päivitetään s.e. se vastaa entistä paremmin havaintoa. Hävinneitä vektoreita ei päivitetä lainkaan tai päivitetään pienemmällä opetuskertoimella kuin voittajavektoria

Kilpailuun perustuvan klusterointialgoritmin yleistetty muoto ('General Competitive Learning Scheme' GCLS):

- Alustus: $t = 0$, $m = m_{\text{init}}$ (alkuarvaus klustereiden lkm:lle),
  (A) kaikkien tarvittavien parametrien alustus mm vektorit $\mathbf{w}_1, \ldots, \mathbf{w}_m$
- Toista kunnes vektorit $\mathbf{w}_1, \ldots, \mathbf{w}_m$ eivät enää muutu merkittävästi tai

$t = t_{max}$ (yläraja iteraatioaskelien lkm:lle)

- $t = t + 1$
- Poimi satunnainen havainto $\mathbf{x} \in X$
- (B) Etsi havaintoa $\mathbf{x}$ parhaiten vastaava vektori $\mathbf{m}_j$
- (C) Jos ($\mathbf{x}$ ja $\mathbf{w}_j$ eivät ole riittävän samankaltaiset) TAI (jokin muu ehto) JA ($m < m_{\mathsf{max}}$) (yläraja klustereiden lkm:lle), päivitä $m = m + 1$ ja määritä $\mathbf{w}_m = \mathbf{x}$
- (D) Muuten, päivitä vektorit $\mathbf{w}_1, \ldots, \mathbf{w}_m$ seuraavasti:

$$\mathbf{w}_j(t) = \begin{cases} \mathbf{w}_j(t-1) + \eta h(\mathbf{x}, \mathbf{w}_j(t-1)), & \text{jos } \mathbf{w}_j \text{ on voittaja} \\ \mathbf{w}_j(t-1) + \eta' h(\mathbf{x}, \mathbf{w}_j(t-1)), & \text{jos } \mathbf{w}_j \text{ ei ole voittaja} \end{cases} \tag{218}$$

(η ja η' ovat opetuskertoimia ja $h(\cdot, \cdot)$ on jokin ongelmaan sopiva funktio)

Useimmat kilpailuun perustuvat klusterointialgoritmit voidaan esittää GCLS-tyyppisinä, kunhan kohdissa (A)-(D) tehdään sopivat valinnat

Oppiva vektorikvantisaatio

Oppiva vektrorikvantisaatio ('Learning Vector Quantization', LVQ) on esimerkki GCLS-tyyppisestä klusterointialgoritmistä

LVQ-algoritmissä oletetaan klustereiden lkm $m = m_{\mathsf{init}} = m_{\mathsf{max}}$ tunnetuksi

LVQ-algoritmi saadaan kilpailuun perustuvien klusterointialgoritmien yleistetystä muodosta, kun

- (A) Alustetaan vektorit $\mathbf{w}_1, \ldots, \mathbf{w}_m$ satunnaisesti (ei muita parametrejä!)
- (B) $\mathbf{w}_j$ on havaintoa $\mathbf{x}$ parhaiten vastaava voittajavektori, jos $d(\mathbf{x}, \mathbf{w}_j) = \min_{k=1,\ldots,m} d(\mathbf{x}, \mathbf{w}_k)$. Funktio $d(\cdot, \cdot)$ on vektoreiden välinen etäisyys
- (C) Ehto ei toteudu koskaan, koska klustereiden lkm on kiinnitetty

- (D) Vektoreita päivitetään seuraavasti:

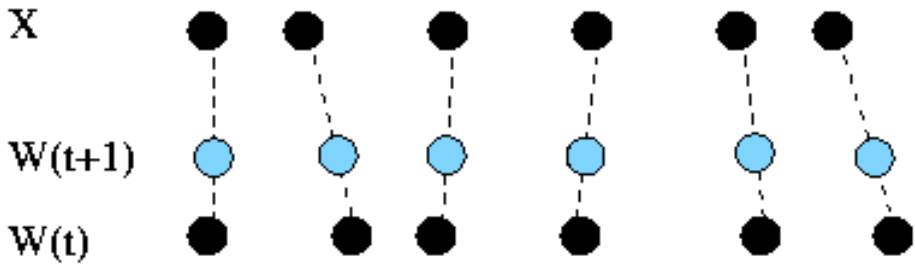
$$\mathbf{w}_j(t) = \begin{cases} \mathbf{w}_j(t-1) + \eta(\mathbf{x} - \mathbf{w}_j(t-1)), & \text{jos } \mathbf{w}_j \text{ on voittaja} \\ \mathbf{w}_j(t-1), & \text{muuten} \end{cases} \tag{219}$$

Jos havaintojen luokat tunnetaan, voidaan vektorit $\mathbf{w}_1, \ldots, \mathbf{w}_m$ varata tietyille luokille ja tehdä päivitys seuraavasti:

$$\mathbf{w}_j(t) = \begin{cases} \mathbf{w}_j(t-1) + \eta(\mathbf{x} - \mathbf{w}_j(t-1)), & \text{jos } \mathbf{w}_j \text{ on oikea voittaja} \\ \mathbf{w}_j(t-1) - \eta(\mathbf{x} - \mathbf{w}_j(t-1)), & \text{jos } \mathbf{w}_j \text{ on väärä voittaja} \\ \mathbf{w}_j(t-1), & \text{muuten} \end{cases} \tag{220}$$

(η on opetuskerroin)

LVQ-esimerkki: voittajavektori $\mathbf{w}(t)$ päivitetään vektoriksi $\mathbf{w}(t+1)$ havainnon $\mathbf{x}$ perusteella:



Itseorganisoituva kartta

('Self-Organizing Map' SOM)

SOM-algoritmissä havainnot yritetään esittää hilaan (kartta) järjestettyjen vektoreiden (karttayksiköiden) $\mathbf{w}_1, \ldots, \mathbf{w}_m$ avulla

SOM-algoritmin tavoitteena on asettaa vektoreiden arvot siten, että ne edustavat mahdollisimman hyvin havaintoja ja että hilassa toisiaan lähellä olevien vektoreiden arvot ovat samankaltaisempia kuin hilassa kaukana toisistaan olevien vektoreiden arvot

SOM-algoritmi saadaan kilpailuun perustuvien klusterointialgoritmien yleistetystä muodosta, kun

- (A) Alustetaan vektorit $\mathbf{w}_1, \ldots, \mathbf{w}_m$ (ei muita parametrejä)
- (B) $\mathbf{w}_{\text{BMU}}$ on havaintoa $\mathbf{x}$ parhaiten vastaava voittajavektori, jos $d(\mathbf{x}, \mathbf{w}_j) = \min_{k=1,\ldots,m} d(\mathbf{x}, \mathbf{w}_k)$. Funktio $d(\cdot, \cdot)$ on vektoreiden välinen etäisyys

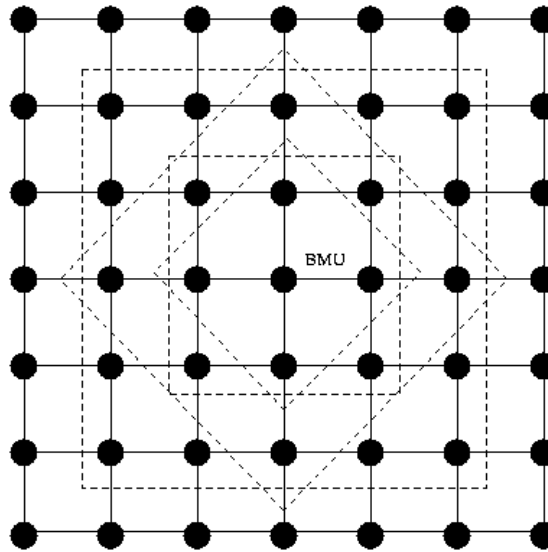
- (C) Ehto ei toteudu koskaan, koska vektoreiden lkm on kiinnitetty
- (D) Päivitä vektoreita seuraavasti:

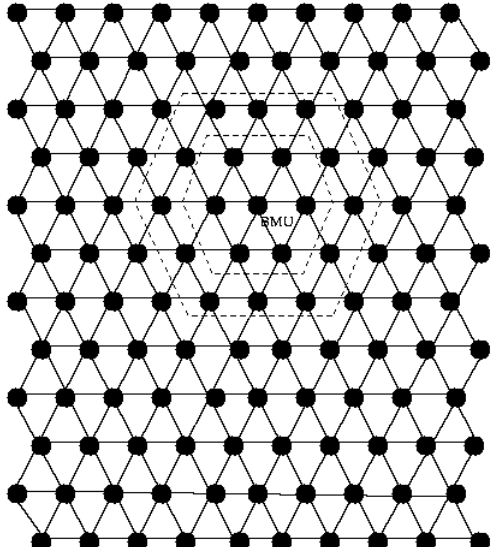
$$\mathbf{w}_j(t) = \mathbf{w}_j(t-1) + \eta h(j, \mathsf{BMU})(\mathbf{x} - \mathbf{w}_j(t-1)), \tag{221}$$

  missä η on opetuskerroin ja $h(\cdot, \cdot)$ on naapurustofunktio, joka määrää mitkä ovat voittajavektorin päivitettäviä naapureita

SOM-algoritmin antamaa tulosta voidaan analysoida ns. U-matriisin (kertoo kuinka samankaltaisia naapurivektorit ovat) ja komponettitasojen (kertovat millasia arvoja piirteet saavat erikohdissa karttaa) avulla

Erilasia 2-ulotteisia hiloja ja naapurustoja:




BMU