SGN-2500: Johdatus hahmontunnistukseen

Jussi Tohka
Tampereen teknillinen yliopisto
Signaalinkäsittelyn laitos
2006 - 2012

24. helmikuuta 2012

Esipuhe

Tämä moniste on syntynyt vuosina 2003 ja 2004 TTY:llä luennoimani kurssin 'Introduction to Pattern Recognition' pohjalta. Varsinkin vuoden 2003 kurssi perustui kirjaan Duda, Hart, Stork: Pattern Classification, 2nd Edition. Kurssi on sittemmin muuttunut löyhemmin kirjaan perustuvaksi, mutta esimerkiksi asioiden käsittelyjärjestys kurssilla ja monisteessa mukailee Dudan, Hartin ja Storkin kirjaa. Käsitteiden suomentamisessa on suureksi avuksi ollut Petri Koistisen 'Tilastollinen hahmontunnistus' moniste Helsingin yliopistosta vuodelta 2002.

Moniste vastaa neljän opintopisteen kurssia, jonka luennointiin käytettiin 24 luentotuntia vuonna 2006. Perusideana on ollut, että perusmateriaali käydään tarkasti läpi. Tämän lisäksi joitakin perusteluja esitetään ylimalkaisempaan tapaan. Tarkoituksena on ollut tarjota yleiskuva tilastollisesta hahmontunnistuksesta ja pohja hahmontunnistuksen opiskelulle jatkossa. Tämä tarkoittaa, että pääpaino on hahmontunnistuksen yleisissä tilastollisissa periaatteissa eikä eri hahmontunnistusmenetelmien ja niiden sovellusten esittelyssä.

Käsillä oleva moniste on korjattu versio vuoden 2006 vastaavasta monisteesta. Korjaukset ovat olleet varsin kosmeettisia liittyen lähinnä monisteen kieliasuun, ja joidenkin monisteessa olleiden virheiden oikaisuun. Oikolukemisessa on suureksi avuksi ollut tekn. yo. Juho Kunnari. Keskustelut Jari Niemen ja Mikko Parviaisen kanssa ovat auttaneet selventämään monisteen tiettyjä osia.

Tampere syksyllä 2008

Jussi Tohka

Sisältö

1 Johdanto **1**

2 Hahmontunnistusjärjestelmät **3**
2.1 Esimerkkejä . . . 3
2.1.1 Optinen merkintunnistus eli OCR . . . 3
2.1.2 Fisherin/Andersonin iirisdata . . . 3
2.2 Hahmontunnistusjärjestelmän perusrakenne . . . 3
2.3 Hahmontunnistusjärjestelmän suunnittelu . . . 7
2.4 Ohjattu ja ohjaamaton oppiminen . . . 7

3 Todennäköisyyslaskentaa **9**
3.1 Esimerkkejä . . . 9
3.2 Perusmääritelmät ja terminologiaa . . . 10
3.3 Todennäköisyysavaruuksien perusominaisuuksia . . . 11
3.4 Satunnaismuuttujat ja satunnaisvektorit . . . 11
3.5 Tiheys ja kertymäfunktiot . . . 13
3.5.1 Kertymäfunktio . . . 13
3.5.2 Tiheysfunktio: Diskreetit satunnaismuuttujat . . . 13
3.5.3 Tiheysfunktio: Jatkuvat satunnaismuuttujat . . . 14
3.6 Ehdollinen todennäköisyys ja tilastollinen riippumattomuus . . . 16
3.6.1 Tapahtumat . . . 16
3.6.2 Satunnaismuuttujat . . . 17
3.7 Bayesin sääntö . . . 18
3.8 Odotusarvo ja varianssi . . . 20
3.9 Moniulotteinen normaalijakauma . . . 21

4 Bayesin päätösteoria ja optimaaliset luokittimet **25**
4.1 Luokitusongelma . . . 25
4.2 Luokitusvirhe . . . 26
4.3 Bayesin minimivirheluokitin . . . 27
4.4 Bayesin minimiriskiluokitin . . . 28
4.5 Erotinfunktiot ja päätöspinnat . . . 30
4.6 Erotinfunktiot normaalijakautuneille luokille . . . 32
4.7 Riippumattomat binääripiirteet . . . 34

5 Ohjattu luokkatiheysfunktioiden oppiminen 41
5.1 Ohjattu oppiminen ja Bayes luokitin . . . 41
5.2 Tiheysfunktion parametrien estimointi . . . 42
5.2.1 Parametrien estimoinnin idea . . . 42
5.2.2 Suurimman uskottavuuden estimaatti . . . 42
5.2.3 ML-estimaatin löytäminen . . . 43
5.2.4 ML-estimaatit normaalijakaumalle . . . 43
5.2.5 ML-estimaatin ominaisuuksia . . . 44
5.2.6 Luokitusesimerkki . . . 45
5.3 Tiheysfunktion epäparametrinen estimointi . . . 46
5.3.1 Histogrammi . . . 46
5.3.2 Tiheyden estimoinnin yleinen formulointi . . . 47
5.4 Parzen ikkunat . . . 49
5.4.1 Histogrammeista Parzen ikkunoihin . . . 49
5.4.2 Tiheysfunktion Parzen-estimaatti . . . 49
5.4.3 Parzen-estimaattien ominaisuuksia . . . 50
5.4.4 Ikkunanleveyden valinta . . . 51
5.4.5 Parzen-estimaatit luokituksessa . . . 51
5.4.6 Tilastolliset neuroverkot . . . 52
5.5 k:n lähimmän naapurin menetelmä . . . 53
5.5.1 k_n lähimmän naapurin tiheysestimaatit . . . 53
5.5.2 Lähimmän naapurin sääntö . . . 54
5.5.3 k_n lähimmän naapurin sääntö . . . 54
5.5.4 Metriikoita . . . 56
5.6 Virhelähteistä . . . 57

6 Lineaariset erotinfunktiot ja luokittimet 59
6.1 Johdanto . . . 59
6.2 Lineaaristen luokittimien ominaisuuksia . . . 60
6.2.1 Lineaariset luokittimet . . . 60
6.2.2 Kaksi luokkaa . . . 60
6.2.3 Useampi luokka . . . 61
6.3 Lineaarisesti separoituvat harjoitusnäytteet . . . 62
6.4 Perceptron kriteeri ja algoritmi kahden luokan tapauksessa . . . 64
6.4.1 Perceptron kriteeri . . . 64
6.4.2 Perceptron algoritmi . . . 65
6.5 Perceptron useammalle luokalle . . . 66
6.6 Pienimmän neliösumman kriteeri . . . 67

7 Luokitinten testaus 71
7.1 Virhetodennäköisyyden arviointi . . . 71
7.2 Sekaannusmatriisi . . . 72
7.3 Esimerkki . . . 72

8 Ohjaamaton luokitus eli klusterointi **75**
8.1 Johdanto 75
8.2 Klusterointiongelma 76
8.3 K-means klusterointi 76
8.3.1 K-means kriteeri 76
8.3.2 K-means algoritmi 77
8.3.3 K-means klusteroinnin ominaisuuksia 77
8.4 Sekoitemallit 79
8.5 EM algoritmi 81

Luku 1

Johdanto

Hahmontunnistus (engl. Pattern recognition) tarkoittaa karkeasti ottaen objektin, tai kohteen, sijoittamista oikeaan luokkaan objektista tehtyjen havaintojen perusteella, yleensä automaattisesti tietokoneen avulla. Tunnistettavat objektit, niistä tehdyt mittaukset ja objektiluokat voivat olla melkein mitä tahansa, joten eri hahmontunnistustehtävät poikkeavat radikaalisti toisistaan. Pullonpalautusautomaatti on yksinkertainen esimerkki hahmontunnistusjärjestelmästä: Automaatin tehtävänä on ottaa vastaan asiakkaiden pullot, tölkit ja juomakorit, tunnistaa ne tiettyyn panttiluokkaan kuuluvaksi ja antaa asiakkaalle hyvitys rahana. Toinen, nykyään arkipäiväinen, esimerkki on roskapostin automaattinen tunnistaminen ja lajittelu eri postilaatikkoon kuin hyödyllinen sähköposti. Hahmontunnistuksella onkin valtava määrä erilaisia sovelluksia lääketieteestä teollisuusautomaatioon.

Joskus hahmontunnistussovellukset ovat hyvinkin arkipäiväisiä ja taas joskus ne ovat hyvinkin kaukana arkipäivästä. Sovellusten arkipäiväisyys, eli se että ihmiset ratkaisevat samankaltaisia ongelmia helposti, ei suinkaan tarkoita, että niihin liittyvät hahmontunnistusongelmat olisivat helppoja. Esimerkiksi tietokonetta on äärimmäisen vaikea opettaa 'lukemaan' käsinkirjoitettua tekstiä. Osa vaikeudesta seuraa siitä, että sama kirjain eri henkilöiden kirjoittamana voi näyttää hyvinkin erilaiselta. Tästä syystä on usein hyödyllistä mallintaa vaihtelua saman luokan (esimerkiksi käsinkirjoitetut A:t) sisällä. Tällä kurssilla perehdymmekin niin sanottuun tilastolliseen hahmontunnistukseen, joka perustuu vaihtelun mallintamiseen tilastollisin menetelmin. Tilastollinen hahmontunnistus jakautuu tällä kurssilla kahteen alaluokkaan. Karkeasti ottaen toisessa mallinnetaan vaihtelua luokkien sisällä, ja toisessa luokkien välistä vaihtelua. Tilastollinen hahmontunnistus, laajasti ymmärrettynä, kattaa suurimman osan kaikista hahmontunnistussovelluksista.

Toinen hahmontunnistuksen pääluokka on nimeltään syntaktinen hahmontunnistus, johon emme tällä kurssilla perehdy. Se perustuu oletukselle, että hahmot (tai paremminkin niistä tehdyt havainnot) jakautuvat aina vain yksinkertaisempiin ja yksinkertaisimpiin alihahmoihin. Hahmontunnistuksesta tulee näin yksinkertaisimpien alkeishahmojen ja niiden suhteen kielen tutkimusta, joka nojaa paljolti formaalien kielten teoriaan. Jotkut tutkijat mielellään erottavat kolmannen alahaaran, nimittäin neuroverkkojen avulla tapahtuvan hahmontunnistuksen. Kuitenkin tämä voidaan myös laskea tilastollisen hahmontunnistuksen piiriin kuuluvaksi. Neu-

roverkkoja käsitellään kursseilla 'Pattern recognition' ja 'Neural computation'.

Luku 2

Hahmontunnistusjärjestelmät

2.1 Esimerkkejä

2.1.1 Optinen merkintunnistus eli OCR

Optisessa merkintunnistuksessa pyritään tunnistamaan automaattisesti kirjainmerkkejä kuva-syötteen perusteella. Englanniksi optista merkintunnistusta kutsutaan nimellä optical character recognition eli OCR. OCR-ongelmat jakautuvat eri vaikeusasteisiin sen mukaan ovatko tunnistettavat merkit käsin kirjoitettuja vai koneellisesti tuotettuja. Sen lisäksi muita ongelman vaikeuteen vaikuttavia seikkoja ovat esimerkiksi kuvasyötteen laatu, voidaanko merkkien olettaa sijaitsevan omissa laatikoissaan (kuten lomakkeissa), ja kuinka monta merkkiä on tarpeellista tunnistaa.

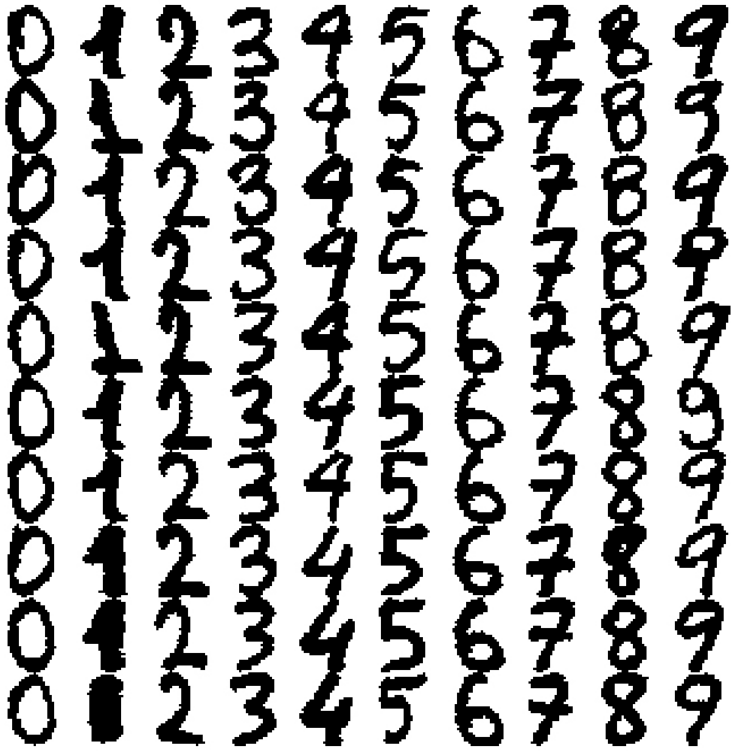
Tarkastelemme kursiivisesti tämän kurssin aikana käsinkirjoitettujen numeroiden tunnistusta. Data tähän tarkasteluun on vapaasti saatavissa University of California Irvinen koneoppimistietokannasta ja sen ovat sinne lahjoittaneet E. Alpaydin ja C. Kaynak Istanbulin yliopistosta. Esimerkkejä käsinkirjoituista merkeistä on kuvassa 2.1.

2.1.2 Fisherin/Andersonin iirisdata

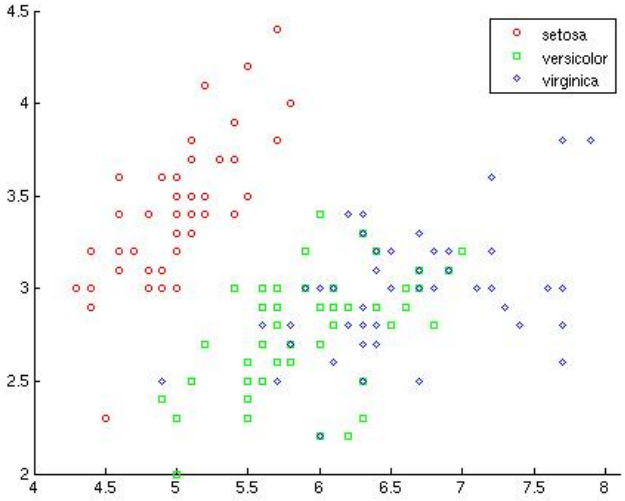
Ehkä kuuluisin luokittimien vertailussa käytetty aineisto on Edgar Andersonin vuonna 1935 keräämä aineisto kolmen iirislajikkeen (kurjenmiekkalajikkeen) terä- ja verholehtien pituuksista ja leveyksistä. R.A. Fisher teki aineiston kuuluisaksi käyttämällä sitä esimerkkinä vuonna 1936 julkaistussa artikkelissa 'The use of multiple measurements in taxonomic problems', jota voidaan pitää ensimmäisenä avauksena hahmontunnistukseen. Kuvassa 2.2 on näytetty verholehtien pituudet ja leveydet aineistossa.

2.2 Hahmontunnistusjärjestelmän perusrakenne

Hahmontunnistusjärjestelmän tarkoituksena on sijoittaa jokin objekti sitä esittävään luokkaan objektista olevan mittausdatan perusteella. Monet hahmontunnistusjärjestelmät voidaan jakaa viiteen vaiheeseen:



Kuva 2.1: Käsinkirjoitettuja merkkejä, 10 esimerkkiä kustakin merkistä. Kaikki merkit esitetään 32×32 binäärikuvana.



Kuva 2.2: Fisherin iiristen verholehtien leveydet (x-akseli) ja pituudet (y-akseli).

1. Mittaus;
2. Esikäsittely ja segmentointi;
3. Piirteenirrotus;
4. Luokitus;
5. Jälkikäsittely;

Menetelmät ja välineistö suurimmalle osalle näistä vaiheista riippuvat paljolti sovelluksesta. Esimerkiksi puheentunnistuksen ja sormenjälkien tunnistuksen vaatimat mittaukset ovat varsin erilaisia. Samoin ovat mittaukset, joita tarvittiin OCR:ssä, ja Anderssonin iiriksistä tekemät mittaukset. Luokituksen tueksi voidaan esittää kuitenkin varsin kattava teoria ja valikoima yleisiä algoritmeja, joten keskitymmekin tällä kurssilla luokittimien suunnitteluun tilastollisia periaatteita hyödyntäen. Kuitenkin lopullinen päämäärä on suunnitella hahmontunnistusjärjestelmä ja sen takia on tärkeää ymmärtää kaikkien näiden vaiheiden merkitys, vaikka emme useimpiin pystykkään sovelluksesta riippumattomia työkaluja antamaan.

Mittaus tarkoittaa tässä jollakin tekniikalla luokiteltavasta kohteesta mitattavaa dataa, esimerkiksi kuvaa tai ääntä. Usein yksi mittaus sisältää informaatiota useista luokiteltavista objekteista vaikkakin luokituksen kannalta ne on hyvä nähdä erillisinä objekteina. Esimerkkinä toimikoon automaattinen osoitteentunnistus, jossa siis täytyy tunnistaa useita kirjainmerkkejä koko osoitteen selville saamiseksi. Tässä tapauksessa mittaus sisältää luultavasti kuvan koko joukosta kirjainmerkkejä, sekä mahdollisesti taustaa jolla ei ole mitään tekemistä itse osoitteentunnistustehtävän kannalta.

Esikäsittely tarkoittaa suodatusta kohinan vähentämiseksi mittausdatasta ja muita esikäsittelyoperaatioita. **Segmentointi** tarkoittaa mittausdatan jakamista yhtä luokitettavaa objektia kuvaaviin osiin. Esimerkiksi osoitetta tunnistettaessa koko osoitteen kuva pitää jakaa yhden merkin kuvaa vastaaviin kuvapaloihin. Segmentoinnin tuloksena olevaa (digitaalista) esitystä kohteesta vektorina kutsutaan *hahmovektoriksi* (engl. pattern vector).

Piirteenirrotus (engl. feature extraction). Erityisesti kuvallista informaatiota käsiteltäessä mittausdataa yhtäkin objektia kohden on paljon. Korkearesoluutioisessa kasvokuvassa voi olla 1024*1024 eli yli miljoona pikseliä, josta voi päätellä että esimerkiksi kasvojentunnistuksessa hahmovektorit elävät erittäin korkeaulotteisessa avaruudessa. Kaikki tästä datasta ei ole hyödyllistä luokitusta ajatellen. Piirteenirrotuksen tehtävä onkin löytää sellaiset piirteet, joiden perusteella luokitus voi parhaiten tapahtua. Näitä luokittimelle annettavia piirteitä kutsutaan nimellä *piirrevektori* (engl. feature vector) ja kaikkien piirrevektorien joukkoa taas kutsutaan *piirreavaruudeksi*. Kasvojentunnistuksessa piirteiden määrää rajoitetaankin usein pääkomponenttianalyysin avulla. Matemaattisten tekniikoiden lisäksi piirteenirrotusta voi tehdä heuristisesti valitsemalla sovelluksesta riippuvia piirteitä joiden aavistaa olevan hyödyllisiä luokituksessa. Joskus taas dataa kohdetta kohden on vain

rajallinen määrä, jolloin on luonnollista valita piirrevektoriksi suoraan hahmovektori. Näin on esimerkiksi usein hahmontunnikseen perustuvassa kuvien segmentoinnissa. Piirteenirrotus onkin usein hyvin sovelluksesta riippuvaa, tosin joitain yleisiä matemaattisia tekniikoita piirteenirrotukseen ja valintaan on olemassa. Näistä lisää kurssilla 'Pattern recognition'.

Yleisesti piirteenirrotuksen ja luokituksen ero on varsin häilyväinen, ainakin sovelluksen kannalta. Piirteenirrottimen tehtävänä tuottaa datasta esitys, joka on mahdollisimman helppo luokittaa. Mikä esitys on sitten helppo luokittaa riippuu taas käytettävästä luokittimesta ja siispä mitään universaalisti optimaalisia piirteitä ei ole olemassa.

Luokitin (engl. classifier) ottaa syötteenään kohteesta irrotetun piirrevektorin ja asettaa kohteen piirrevektorin perusteella sopivimpaan luokkaan. Tällä kurssilla keskitytään luokittimien suunnittelun opiskeluun. Tämä siksi, että piirrevektoriabstraktio mahdollistaa pitkälti sovelluksesta riippumattoman teorian luokittimien suunnittelua varten.

Yleisesti luokitusongelma pyritään ratkaisemaan siten, että odotettu luokitusvirhe on mahdollisimman pieni, eli mahdollisimman moni **kaikista mahdollisista** piirrevektoreista luokitetaan oikein. Tämä on haastavaa, koska saman luokan piirrevektoreissa esiintyy variaatiota (ks. kuva 2.3). Tätä variaatiota on helpoin mallintaa todennäköisyyslaskennan perusteella. Seuraavassa luvussa kertaammekin todennäköisyyslaskentaa, ja luvussa 4 esitämme todennäköisyyslaskentaan perustuvan luokitusteorian.

Kuva 2.3: Vasemmalta: Kaksi esimerkkiä käsinkirjoitetuista kakkosista ja oikealla nämä esitettynä päällekkäin siten, että valkoiset pikselit ovat niitä jotka ovat valkoisia kummassakin esimerkissä, harmaat niitä jotka ovat valkoisia jommassa kummassa esimerkissä, ja mustat niitä, jotka ovat mustia kummassakin esimerkissä.

Jälkikäsittely Usein hahmontunnistusjärjestelmän lopullinen tehtävä on päättää toimenpiteestä luokittimista saatujen tulosten perusteella. Jos luokitustuloksia on useita, tässä voidaan käyttää hyväksi niiden keskinäisiä riippuvuussuhteita, eli *kontekstia*. Esimerkiksi osoitteen tunnistuksen yhteydessä ei tällöin yhden merkin tunnistaminen väärin välttämättä tarkoita, että koko osoite tunnistettaisiin väärin. Tarkemmin, jos luokituksen tuloksena on 'Hämeenketu' voidaan olettaa oikean kadunnimen olevan 'Hämeenkatu'.

2.3 Hahmontunnistusjärjestelmän suunnittelu

Hahmontunnistusjärjestelmän suunnittelu on pitkälti iteratiivinen prosessi. Jos järjestelmä ei ole tarpeeksi luotettava parannetaan yhtä tai useampaa järjestelmän viidestä vaiheesta (katso edellinen kappale 2.2). Sen jälkeen testataan järjestelmä uudestaan ja parannetaan sitä, kunnes järjestelmä on tarpeeksi hyvä. Tämä iteraatio perustuu sovelluksesta olevaan tietoon, hahmontunnistuksen teorialle ja yritykseen ja erehdykseen. Yrityksen ja erehdyksen osuutta voi minimoida hahmontunnistuksen teoreettisen pohjan tuntemuksella ja yhdistämällä sen sovellusalueesta olevaan tietoon.

Milloin järjestelmä on sitten tarpeeksi luotettava? Tämä tietenkin riippuu sovelluksen tarpeista ja järjestelmän luotettavuuden mittaamista käsitellään tarkemmin kurssin loppupuolella. Järjestelmän luotettavuutta voidaan parantaa esimerkiksi lisäämällä piirrevektoreihin uusi piirre, joka on mahdollisimman riippumaton jo käytössä olevista piirteistä. Aina tämä ei kuitenkaan ole mahdollista, esimerkiksi Fisherin iirisaineiston kanssa piirteiden määrä oli rajoitettu neljään, koska alkuperäisiin mittauksiin ei päästy vaikuttamaan. Toinen vaihtoehto on parantaa luokitinta. Tähänkin on useita erilaisia keinoja, joista ei aina ole yksinkertaista valita parasta. Tarkastelemme tätäkin dilemmaa hiukan myöhemmin, kunhan tarvittavat pohjatiedot luokittimista ovat kasassa.

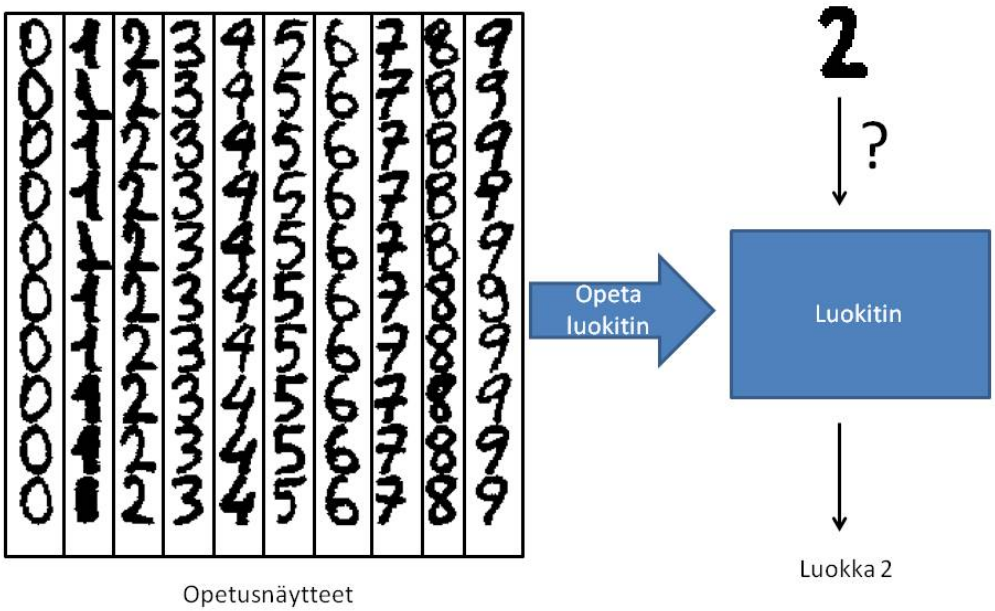
Muita järjestelmän suunnittelun kannalta tärkeitä asioita ovat esimerkiksi järjestelmän laskennallinen kompleksisuus ja sen vaatiman mittauslaitteiston hinta.

2.4 Ohjattu ja ohjaamaton oppiminen

Hahmontunnistusjärjestelmän luokitinkomponentti täytyy opettaa tunnistamaan tietyt piirrevektorit tiettyyn luokkaan kuuluviksi. Oppiminen (tai opettaminen) voidaan tässä tapauksessa jakaa kahteen päätyyppiin, nimittäin *ohjattuun oppimiseen* ja *ohjaamattomaan oppimiseen*. Hahmontunnistuksessa (ja tässä monisteessa) käytetään myös termejä *ohjattu luokitus* ja *ohjaamaton luokitus*.

Ohjatussa luokituksessa (engl. supervised classification) meillä on käytössämme esimerkkejä, jotka sisältävät sekä mittausdatan (piirrevektorin tai hahmovektorin) kohteesta, että tiedon kohteen luokasta, ks. kuva 2.4. Nämä esimerkit, joita kutsutaan myös *prototyypeiksi* tai *opetusnäytteiksi*, joudutaan usein sijoittamaan luokkiin manuaaliasesti. Tämä voi olla työlästä: juuri tämän välttämiseksi hahmontunnistusjärjestelmä alunperin päätettiin suunnitella. Niinpä esimerkkejä on yleensä kovin vähän verrattuna mahdollisiin järjestelmän saamiin syötteisiin. Eli, siis ongelmana on prototyyppien perusteella ennustaa ennennäkemattömän piirrevektorin luokka. Tämän takia luokittiminen suunnittelun täytyy perustua prototyyppien lisäksi hahmontunnistusongelman luonteesta tehtyihin oletuksiin. Nämä oletukset voidaan usein parhaiten ilmaista tilastollisesti.

Ohjaamattomassa luokituksessa (engl. unsupervised classification) eli *klusteroinnissa* (engl. clustering) opetusnäytteitä ei ole saatavissa. Tällöin piirrevektorien luokituksen täytyy perustua niiden väliseen samankaltaisuuteen, jonka perusteella pi-



Kuva 2.4: Ohjattu luokitus

irrevektorit jaetaan 'luonnollisiin ryhmiin'. Mitkä piirrevektorit ovat samankaltaisia riippuu sovelluksesta. Ohjaamaton luokitus on luonnollisesti kertaluokkaa vaikeampi ongelma kuin ohjattu luokitus. Kuitenkin on tilanteita, joissa ohjattu luokitus ei yksinkertaisesti ole mahdollista ja näin ollen on välttämätöntä turvautua ohjaamattomaan luokitukseen. Esimerkiksi tämä on tilanne, jos syötedatan voidaan olettaa muuttuvan ajan kuluessa.

Esittelemme vielä kolmannen oppimistyypin, nimittäin *vahvistusoppimisen* (engl. reinforcement learning). Tässä oppimistyypissä ei ole käytössä valmiiksi luokiteltuja oppimisnäytteitä, mutta opettaja antaa luokitustuloksesta palautteen, joka kertoo onko automaattisen järjestelmän tekemä luokitus oikea vai väärä. Esimerkiksi optisessa merkintunnistuksessa hahmontunnistusjärjestelmän luokitus piirrevektorille on 'R'. Oikea luokitus sen sijaan olisi ollut 'B'. Vahvistusoppimisessa luokitin saa vain tiedon, että se on ollut väärässä, eikä siis tietoa oikeasta luokasta. Tämän enempää emme vahvistusoppimista tällä kurssilla käsittele.

Luku 3

Todennäköisyyslaskentaa

Todennäkoisyyslaskennan perusteiden ymmärtäminen on välttämätöntä hahmontunnistuksen tilastollisten perusteiden ymmärtämiselle. Onhan hahmontunnistusmenetelmien tarkoituksena johtaa luokitin, joka tekee mahdollisimman vähän virheitä. Luokitusvirheitä taas analysoidaan tilastollisesti, koska tiettyä luokkaa kuvaavissa piirrevektoreissa esiintyy tilastollista vaihtelua. Kovin pitkälle meneviä todennäköisyyslaskennan tuloksia ei hahmontunnistuksen johdantokurssilla tarvita. Mutta peruskäsitteiden, kuten satunnaisvektori, on hyvä olla hanskassa. Tämän luvun tarkoituksena onkin kerrata todennäköisyyslaskennan perusteet.

Tärkeintä tässä luvussa on ymmärtää todennäköisyyden konsepti tietynasteisen matemaattisen abstraktion tasolla. Tämä on välttämätöntä, jos halutaan luoda, arvioida, valita, ja soveltaa erilaisia oppimisalgoritmeja. Tietokone ei ymmärrä käsien heiluttelua, vaan sille täytyy antaa tarkat ohjeet (algoritmi) kuinka suoriutua annetusta hahmontunnistustehtävästä.

3.1 Esimerkkejä

Koskapa todennäköisyyslaskenta alkujaan syntyi uhkapelin tarpeista, aloitamme traditioon hyvin istuvilla esimerkeillä.

Mikä on todennäköisyys, että täydestä ja sekoitetusta korttipakasta vedetty kortti on ässä?

Yleisesti ottaen *tapahtuman* (engl. event) todennäköisyys on suotuisien tapausten määrä jaettuna kaikkien mahdollisten tapausten kokonaismäärällä. Tässä tapauksessa suotuisia tapauksia ovat kaikki ässät, joita on neljä. Kortteja taas on yhteensä 52, joten tapahtuman todennäkoisyydeksi tulee 4/52 eli 1/13.

Ajatellaan seuraavaksi kahden nopan yht'aikaista heittämistä. Eri lopputuloksia tässä on yhteensä 36 kappaletta. Millä todennäköisyydellä silmälukujen summa on yhtä kuin 5? Taas jaetaan suotuisien tapausten määrä (4) kaikkien tapausten määrällä (36). Joten vastaus on 1/9.

3.2 Perusmääritelmät ja terminologiaa

Sanomme joukkoa S satunnaiskokeen *otosavaruudeksi*, jos jokainen mahdollinen satunnaiskokeen tulos liittyy yksikäsitteiseen joukon S alkioon. Näitä alkioita kutsutaan *alkeistapauksiksi*.

Tapahtuma on S:n osajoukko, toisin sanoen tapahtuma on joukko satunnaiskokeen tulosvaihtoehtoja. Jos kahden tapahtuman leikkaus on tyhjä, niin ne ovat *toisensa poissulkevat*. Yleisemmin, tapahtumat $E_1, E_2, \ldots, E_n$ ovat toisensa poissulkevia, jos $E_i \cap E_j = \emptyset$ kaikille $i \neq j$. Toisensa poissulkevat tapahtumat - niinkuin nimikin kertoo - ovat sellaisia, että toisen realisoituminen estää toisen realisoitumisen.

Todennäköisyysmitta liittää todennäköisyyden jokaiseen otosavaruuden tapahtumaan, siis se ilmaisee kuinka yleisesti jokin tapahtumaan kuuluva tulosvaihto tulee tulokseksi satunnaiskokeessa. Tarkemmin, *todennäköisyysmitta otosavaruudessa* S on reaaliarvoinen funktio P, joka on määritelty S:n tapahtumille ja joka täyttää seuraavat Kolmogorovin aksioomat:

1. Kaikille tapahtumille $E \subseteq S$, $0 \leq P(E) \leq 1$.

2. $P(S) = 1$.

3. Jos tapahtumat $E_1, E_2, \ldots, E_i, \ldots$ ovat toisensa poissulkevia, niin

$$P(\bigcup_{i=1}^{\infty} E_i) = P(E_1) + P(E_2) + \cdots + P(E_i) + \cdots$$

Kolmogorovin aksioomilla on luontevat tulkinnat (ns. frekvenssitulkinnat): Aksiooma 1 vastaa esiintymisfrekvenssille reaalilukuna luontevia ehtoja. Negatiiviset tai yhtä suuremmat (eli suuremmat kuin 100%) esiintymisfrekvenssit eivät ole järkeviä. Aksiooma 2 taas kertoo, että joka kokeessa tapahtuu otosavaruuteen kuuluva tapahtuma. Aksiooma 3 puolestaan kertoo, että jos tapahtumat ovat toisensa poissulkevia, eivätkä siis voi tulla tulokseksi samassa kokeessa, niin todennäköisyys, että joku tapahtumista toteutuu on yhtä suuri kuin tapahtumien todennäköisyyksien summa. Huomaa, että toistensa poissulkevuus on välttämätöntä, jotta tässä olisi järkeä. [1]

Esimerkki. Korttiesimerkissä luontevan otosavaruuden muodostaa kaikkien korttien joukko. Satunnaiskokeen tuloksena on pakasta vedetty kortti. Meitä kiinnostanut tapahtuma on pakassa olevien ässien joukko. Todennäköisyys kullekkin kortille (alkeistapaukselle) on 1/52 ja muiden tapahtumien todennäköisyydet voidaan laskea tämän perusteella.

[1] Täsmällisesti todennäköisyysavaruus määritellään kolmikkona $(S, \mathcal{F}, P)$, missä $\mathcal{F}$ on tietyt kriteerit täyttävä (ns. sigma-algebra) S:n osajoukkojen joukko, jossa siis P on määritelty. Näin täsmälliset määritelmät ylittävät kuitenkin kurssin tarpeet.

3.3 Todennäköisyysavaruuksien perusominaisuuksia

Edellisessä kappaleessa todennäköisyys määriteltiin otosavaruuden osajoukoille. Määrittelemällä otosavaruus yleisenä joukkona saavutetaan se, että tässä viitekehyksessä johdetut tulokset ovat yleisesti voimassa. Ovathan esimerkiksi sekä $\{0, 1, 2, 3\}$ että reaaliväli $[5.6, 6.99]$ joukkoja. Monia näppäriä perusominaisuuksia voidaankin johtaa joukko-oppiin perustuen. Siksi pieni sanakirja onkin paikallaan:

joukko-oppi	todennäkoisyys
E^c	E ei tapahdu
$E \cup F$	E tai F tapahtuu
$E \cap F$	sekä E että F tapahtuvat

Huomaa myös, että usein $P(E \cap F)$ merkitään $P(E, F)$.
Ja nyt niitä ominaisuuksia:

Lause 1 *Olkoot E ja F mielivaltaisia S:n tapahtumia. Otosavaruuden S todennäköisyysmitalla P on seuraavat ominaisuudet:*

1. $P(E^c) = 1 - P(E)$
2. $P(\emptyset) = 0$
3. *Jos E on F:n osatapahtuma (siis $E \subseteq F$), niin tällöin* $P(F - E) = P(F) - P(E)$
4. $P(E \cup F) = P(E) + P(F) - P(E \cap F)$

Lisäksi, jos tapahtumat $F_1, \ldots, F_n$ ovat toisensa poissulkevia ja $S = F_1 \cup \cdots \cup F_n$, niin pätee

5. $P(E) = \sum_{i=1}^{n} P(E \cap F_i)$.

Kohdan 5 ehdot täyttävää joukkokokoelmaa kutsutaan *otosavaruuden partitioksi*. Huomioi, että tapahtuma F ja sen komplementti F^c muodostavat aina partition. Lauseen 1 todistukset ovat yksinkertaisia ja tässä monisteessa ne sivuutetaan. Niiden totuudenmukaisuudesta on helppo varmentua esimerkiksi *Venn diagrammien avulla*.

3.4 Satunnaismuuttujat ja satunnaisvektorit

Satunnaismuuttujat kuvaavat satunnaiskokeen käsitettä tilastotieteessä. Hahmontunnistuksessa, jossa siis tarkoituksena on asettaa objekti oikeaan luokkaan siitä tehtyjen havaintojen perusteella, havainnot voidaan mieltää satunnaiskokeeksi.

Matemaattisesti satunnaismuuttuja $X : S \to \mathbb{R}$ määritellään kuvauksena otosavaruudelta reaalisuoralle. Satunnaismuuttujan X saamien arvojen joukkoa $\{X(x) : x \in S\}$ kutsutaan sen *arvojoukoksi*.

Esimerkki Kortinvetoesimerkissä sopiva satunnaismuuttuja on esimerkiksi vedettyjen ässien lukumäärää kuvaava funktio, ts. $X : \text{kortit} \to \{0, 1\}$, missä $X(\text{A}CE) = 1$ and $X(\text{C}ARD) = 0$, missä ACE tarkoittaa ässää ja CARD mitä tahansa muuta korttia. Noppaa heitettäessä sopiva satunnaismuuttuja on esimerkiksi kuvaus otosavaruudelta silmälukujen summalle, esimerkiksi $X(1, 3) = 4$ ja $X(5, 5) = 10$.

Esimerkiksi-sanat ovat yllä syystä: Tiettyyn satunnaiskokeeseen ei voida liittää yksikäsitteistä satunnaismuuttujaa, vaan vaihtoehtoja on useita. Toiset vaihtoehdot ovat tietysti luontevampia kuin toiset.

Satunnaisvektorit ovat satunnaismuuttujien yleistyksiä monimuuttuja-tilanteeseen. Matemaattisesti ne määritellään kuvauksina $X : S \to \mathbb{R}^d$, missä d on luonnollinen luku. Hahmontunnistuksessa harvoin pystytään luokittamaan mitään yhden mittauksen/havainnon/piirteen perusteella, joten satunnaisvektorit ovat varsin hyödyllisiä. Arvojoukon määritelmä on luonnollinen laajennus satunnaismuuttujan arvojoukon määritelmästä. Tästä lähtien emme erottele yksiulotteista satunnaismuuttujaa satunnaisvektorista vaan kutsumme molempia satunnaismuuttujiksi, jollei vektoriluonteen korostamiseen ole erityistä tarvetta.

Miten satunnaismuuttujat sitten liittyvät edellisten lukujen todennäköisyysavaruuksiin? Satunnaismuuttujat siirtävät otosavaruudessa määritellyt todennäköisyydet reaaliavaruuden osajoukoille: Satunnaismuuttujan X arvojoukon osajoukon todennäköisyys on yhtä suuri kuin tämän osajoukon alkukuvan todennäköisyys todennäköisyysavaruudessa. Tämä taas puolestaan mahdollistaa todennäköisyyden mallintamisen todennäköisyysjakaumien avulla. **Satunnaismuuttujasta täytyy käytännössä tuntea vain sen jakauma - näitä käsitellään seuraavassa kappaleessa. Satunnaismuuttujan käsite on kuitenkin välttämätön.**

Usein voimmekin 'unohtaa' otosavaruuden ja ajatella todennäköisyyksiä satunnaismuuttujien kautta. Todennäköisyyslaskennassa usein pudotetaan satunnaismuuttujan argumentti merkinnöistä, ts. $P(X \in B)$ tarkoittaa samaa kuin $P(\{s : X(s) \in B, s \in S, B \subseteq \mathbb{R}^d\})$[2].

Huomaa kuitenkin, että satunnaismuuttujan kertaalleen saama arvo tarkoittaa yhden satunnaiskokeen tulosta. Tämä on eri asia kuin satunnaismuuttuja, ja näitä kahta ei pidä missään olosuhteissa sekoittaa, vaikka merkinnät antaisivatkin siihen mahdollisuuden.

[2]Hyvä ja täsmällinen esitys asiasta löytyy esimerkiksi kirjasta E.R. Dougherty: Probability and Statistics for the Engineering, Computing and Physical Sciences.

3.5 Tiheys ja kertymäfunktiot

3.5.1 Kertymäfunktio

Esitellään ensin kätevä merkintä: Vektoreille $\mathbf{x} = [x_1, \ldots, x_d]^T$ ja $\mathbf{y} = [y_1, \ldots, y_d]^T$ merkintä $\mathbf{x} < \mathbf{y}$ tarkoittaa samaa kuin $x_1 < y_1$ ja $x_2 < y_2$ ja $\cdots$ ja $x_d < y_d$. Eli, jos vektorien komponentteja vertaillaan pareittain, niin $\mathbf{x}$:n komponentit ovat kaikki pienempiä kuin $\mathbf{y}$:n komponentit.

Satunnaismuuttujan X kertymäfunktio F_X määritellään kaikille $\mathbf{x}$

$$F_X(\mathbf{x}) = P(X \leq \mathbf{x}) = P(\{s : X(s) \leq \mathbf{x}\}). \tag{3.1}$$

Kertymäfunktio siis mittaa $\mathbf{x}$:ään asti kertyneen todennäköisyyden kokonaismassaa.

Korttiesimerkissämme satunnaismuuttuja sai arvon yksi, jos pakasta vedetty kortti oli ässä ja arvon nolla muutoin. Tällöin X:n kertymäfunktio on $F_X(x) = 0$, kun $x < 0$, $F_X(x) = 48/52$, kun $0 \leq x < 1$, ja $F_X(x) = 1$ muutoin.

Lause 2 *Kertymäfunktiolla F_X on seuraavat ominaisuudet*

1. *F_X on kasvava ts. Jos $\mathbf{x} \leq \mathbf{y}$, niin $F_X(\mathbf{x}) \leq F_X(\mathbf{y})$.*
2. *$\lim_{\mathbf{x}\to-\infty} F_X(\mathbf{x}) = 0$, $\lim_{\mathbf{x}\to\infty} F_X(x) = 1$.*

Väitteiden todistukset ovat todennäköisyysavaruuksien ominaisuuksien helppoja seurauksia. Nämä esitetään tässä tapahtumien ja satunnaismuuttujien yhteyden hahmottamisen helpottamiseksi.

Todistus. 1) Tapahtuma $X \leq \mathbf{x}$ on tapahtuman $X \leq \mathbf{y}$ alitapahtuma. Näin ollen lauseen 1 kohdasta 3 seuraa, että $P(X \leq \mathbf{y}) - P(X \leq \mathbf{x}) = F_X(\mathbf{y}) - F_X(\mathbf{x}) \geq 0$, joten $F_X(\mathbf{x}) \leq F_X(\mathbf{y})$.
2) Tapahtuma $X \leq -\infty = \emptyset$ ja Tapahtuma $X \leq \infty = S$, joten väiteet seuraavat Lauseen 1 kohdasta 2 ja Kolmogorovin toisesta aksioomasta.

3.5.2 Tiheysfunktio: Diskreetit satunnaismuuttujat

Satunnaismuuttuja X on *diskreetti*, jos sen arvojoukko on numeroituva. (Numeroituvan joukon alkiot voidaan kirjoittaa listana $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots$).

Diskreetin satunnaismuuttujan tiheysfunktio p_X määritellään

$$p_X(\mathbf{x}) = \begin{cases} P(X = \mathbf{x}) & \text{jos } \mathbf{x} \text{ kuuluu } X : n \text{ arvojoukkoon} \\ 0 & \text{muulloin} \end{cases}. \tag{3.2}$$

Toisin sanoen tiheysfunktion arvo $p_X(\mathbf{x})$ on tapahtuman $X = \mathbf{x}$ todennäköisyys. Jos satunnaismuuttujalla X on tiheysfunktio p_X niin sanomme usein, että X on jakautunut tiheysfunktion p_X mukaan ja merkitsemme $X \sim p_X$.

Kertymäfuktio voidaan esittää tiheysfuktion avulla:

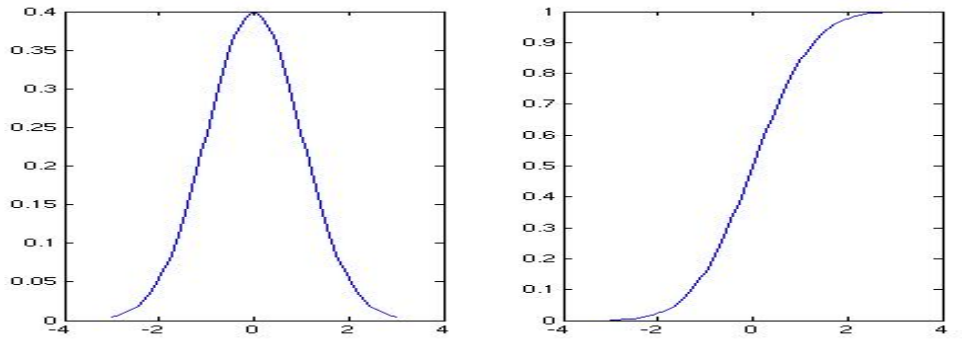
$$F_X(\mathbf{x}) = \sum_{\mathbf{y}\leq\mathbf{x}} p_X(\mathbf{y})$$

3.5.3 Tiheysfunktio: Jatkuvat satunnaismuuttujat

Satunnaismuuttuja X on *jatkuva*, jos on olemassa sellainen funktio p_X, että

$$F_X(\mathbf{x}) = \int_{-\infty}^{\mathbf{x}} p_X(\mathbf{y}) d\mathbf{y}. \tag{3.3}$$

Funktiota p_X kutsutaan tällöin X:n *tiheysfunktioksi*. Kuvassa 3.1 on esitetty normaalijakauman tiheys- ja kertymäfunktiot.



Kuva 3.1: Vasemmalla standardinormaalijakauman tiheysfunktio ja oikealla kertymäfunktio. Tiheysfuktion arvo pistessä x edustaa todennäköisyystiheyttä pisteessä x, esimerkiksi $p_X(0) = 0.4$. Huomaa, ettei tämä itsessään ole todennäköisyys. Sensijaan kertymäfunktiosta voidaan suoraan lukea pisteeseen x asti kertynyt todennäköisyys. Esimerkiksi vasemmasta paneelista nähdään, että $F_X(0) = 0.5$. Tämä tarkoittaa, että tapahtuman $X \leq 0$ todennäköisyys on 0.5. Jos halutaan laskea todennäköisyys tapahtumalle $0 \leq X \leq 0.5$, riittää katsoa kertymä molemmissa loppupisteissä ($F_X(0) = 0.5, F_X(0.5) = 0.6915$), ja kysytty todennäköisyys saadaan $F_X(0.5) - F_X(0) = 0.6915 - 0.5 = 0.1915$

Tällöin todennäköisyys sille, että satunnaismuuttujan X realisaatio $\mathbf{x}$ kuuluu $\mathbb{R}^d$:n osajoukkoon B on

$$P(X \in B) = \int_B p_X(\mathbf{y}) d\mathbf{y}.$$

Jatkuvassa tapauksessa tiheysfunktion arvo $p_X(\mathbf{x})$ yhdessä pisteessä ei ole (itsessään) todennäköisyys. Integraali yhden pisteen yli on kroonisesti nolla, ja niinpä jatkuvassa tapauksessa ei ole järkevää tarkastella tietyn yksittäisen satunnaismuuttujan arvon toteutumisen todennäkoisyyttä.

Huomautuksia integraaleista.

- Voimme tällä kurssilla olettaa, että integraalit ovat normaaleja usean muuttujan Riemann-integraaleja.
- Integraali

$$\int_{-\infty}^{\mathbf{x}} p_X(\mathbf{y}) d\mathbf{y}$$

tarkoittaa samaa kuin integraali

$$\int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_d} p_X(y_1, y_2, \ldots, y_d) dy_1 \cdots dy_d,$$

missä $\mathbf{x} = [x_1, \ldots, x_d]^T$ ja $\mathbf{y} = [y_1, \ldots, y_d]^T$

- Ja lopuksi tärkein kommentti: näitä integraaleja ei hahmontunnistuksessa jouduta käytännössä laskemaan.

Kelvolliset tiheysfunktiot voidaan helposti karakterisoida:

Lause 3 *Funktio $p : \mathbb{R}^d \to \mathbb{R}$ on jonkin jatkuvan satunnaismuuttujan X tiheysfunktio jos ja vain jos*

1. $p(\mathbf{x}) \geq 0$ *kaikille* $\mathbf{x}$.
2. $\int_{-\infty}^{\infty} p(\mathbf{x}) d\mathbf{x} = 1$.

Funktio $p : \mathbb{R}^d \to \mathbb{R}$ on jonkin diskreetin satunnaismuuttujan X tiheysfunktio jos ja vain jos

1. $p(\mathbf{x}) \geq 0$ *kaikille* $\mathbf{x}$.
2. $\sum_{\mathbf{x} \in A} p(\mathbf{x}) = 1$ *missä A on satunnaismuuttujan arvojoukko.*

Ylläoleva lause on suoraa seurausta Kolmogorovin aksioomista. Voidaan lisäksi todeta, että tiheysfunktio määrää yksikäsitteisesti satunnaismuuttujan. Siis tieto satunnaismuuttujan tiheysfunktiosta on usein kaikki mitä tarvitaan.

Esimerkki. Oletetaan, että satunnaismuuttuja on tasaisesti jakautunut välillä $[a, b]$, missä $b > a$. Tällöin sen tiheysfunktio on muotoa

$$p_X(x) = \begin{cases} C & \text{jos} \quad x \in [a, b] \\ 0 & \text{muulloin} \end{cases}$$

missä C on vakio. Miten C tulee valita, jotta p_X olisi tiheysfunktio?

Edellisen lauseen perusteella $\int_{-\infty}^{\infty} p(x) dx = \int_a^b C dx = C(b - a) = 1$. Joten $C = 1/(b - a)$.

Tiheysfunktio on kertymäfunktion derivaatta kunhan vain kertymäfunktio on derivoituva. Vektoritapauksessa tiheysfunktio saadaan kertymäfunktiosta (osittais)derivoimalla se kaikkien vektorimuuttujan komponenttien suhteen.

Kuinka sitten käytännössä löydämme tiettyyn kokeeseen liittyvän satunnaismuuttujan tiheysfunktion? Yksi vaihtoehto on toistaa koetta tarpeeksi monta kertaa, jotta satunnaismuuttujan todennäköisyystiheys saadaan selville. Toinen vaihtoehto on tehdä oletuksia satunnaismuuttujasta. Tämä kysymys on hahmontunnistuksessa erittäin tärkeä, itseasiassa se on generatiivisiin malleihin perustuvan ohjatun oppimisen peruskysymys. Tähän dilemmaan palaamme luvussa 5, missä tarkastelemme eri vaihtoehtoja satunnaismuuttujan tiheysfunktion selvittämiseen, kun joukko esimerkkirealisaatioita on tarjolla.

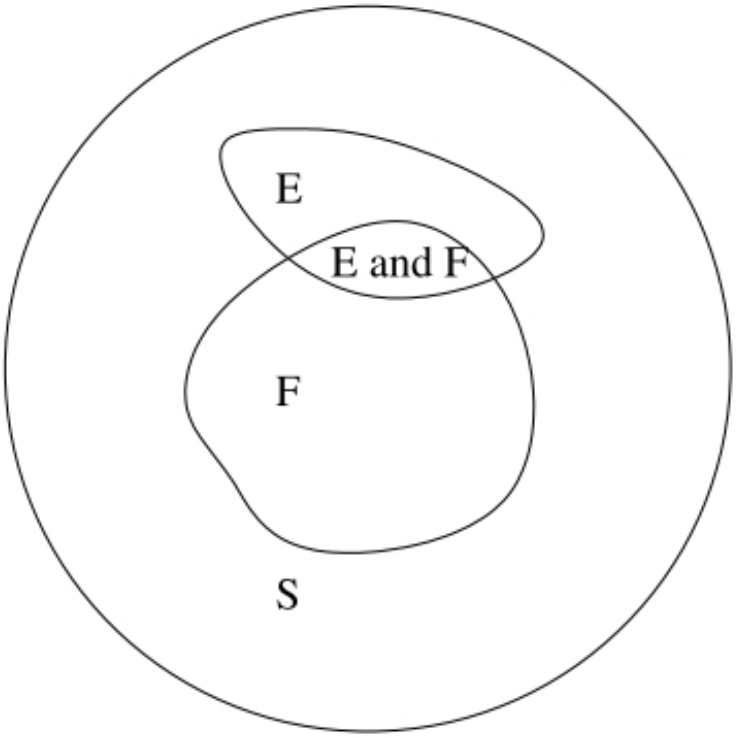
3.6 Ehdollinen todennäköisyys ja tilastollinen riippumattomuus

3.6.1 Tapahtumat

Samaan satunnaismuuttujaan liittyvät tapahtumat $E \subset S$ ja $F \subset S$ ovat *tilastollisesti riippumattomia*, jos

$$P(E \cap F) = P(E)P(F). \tag{3.4}$$

Jos tapahtumat $E \subset S$ ja $F \subset S$ eivät ole tilastollisesti riippumattomia, ne ovat tilastollisesti riippuvia. **Tapahtumien tilastollinen riippumattomuus tarkoittaa, ettei E:n todennäköisyys riipu mitenkään siitä, että realisoituuko F samassa kokeessa vai ei**.



Kuva 3.2: Ehdollisen todennäköisyyden Venn diagrammi.

Oletetaan nyt, että $P(F) \neq 0$. Tapahtuman E *ehdollinen todennäköisyys* tapahtuman F suhteen määritellään (ks. kuva 3.2)

$$P(E|F) = \frac{P(E \cap F)}{P(F)}. \tag{3.5}$$

Tämä tarkoittaa E:n todennäköisyyttä olettaen, että F tiedetään jo realisoituneeksi, eli kokeessa tiedetään tulleen tulokseksi alkeistapahtuma, joka sisältyy tapahtumaan F. Jos E ja F ovat tilastollisesti riippumattomia, niin

$$P(E|F) = \frac{P(E \cap F)}{P(F)} = \frac{P(E)P(F)}{P(F)} = P(E).$$

Vastaavasti, jos $P(E|F) = P(E)$ (eli tapahtuman E todennäköisyys ei riipu tapahtuman F toteumisesta), niin

$$P(E \cap F) = P(E|F)P(F) = P(E)P(F). \tag{3.6}$$

ja tapahtumat E ja F ovat tilastollisesti riippumattomia. **Huomaa myös, että tilastollinen riippumattomuus on eri asia kuin tapahtumien toisensa poissulkevuus.** Jos tapahtumat E ja F ovat toisensa poissulkevat, niin $P(E|F) = P(F|E) = 0$. Tälläiset tapahtumat ovat tilastollisesti riippuvia.

Määritelmä (3.4) ei yleisty suoraviivaisesti vaan tapahtumien $E_1, \ldots, E_n$ tilastollinen riippumattomuus täytyy määritellä induktiivisesti. Määritelmä on tekninen. Merkitään $K = \{i_1, i_2, \ldots, i_k\}$ mielivaltaista indeksijoukon $\{1, \ldots, n\}$ osajoukkoa. Tällöin $E_1, \ldots, E_n$ ovat tilastollisesti riippumattomia, jos

$$P(E_{i_1} \cap \cdots \cap E_{i_k}) = P(E_{i_1}) \cdots P(E_{i_k}), \tag{3.7}$$

kaikille mahdollisille ei-tyhjille osajoukoille $K \subset \{1, \ldots, n\}$. Riippumattomuus on huomattavasti helpompi määritellä satunnaismuuttujien avulla, kuten seuraavassa kappaleessa tullaan havaitsemaan.

Erityisesti kannattaa huomata, että tapahtumien E_1 ja E_2 sekä E_2 ja E_3 sekä E_3 ja E_1 pareittainen tilastollinen riippumattomuus ei takaa tapahtumien E_1, E_2, E_3 tilastollista riippumattomuutta.

3.6.2 Satunnaismuuttujat

Tarkastellaan nyt samaan kokeeseen liittyviä satunnaismuuttujia $X_1, X_2, \ldots, X_n$. Huomaa, että nämä voidaan kirjoittaa myös vektorimuodossa (vektorien vektorina)

$$X = [X_1^T, X_2^T, \ldots, X_n^T]^T.$$

Tiheysfunktiota

$$p_X(\mathbf{x}) = p_{(X_1,\ldots,X_n)}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$$

kutsutaan tällöin satunnaismuuttujien $X_1, X_2, \ldots, X_n$ *yhteistiheydeksi*. Arvo $p_{(X_1,\ldots,X_n)}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ tarkoittaa todennäköisyystiheyttä kun X_1 saa kokeessa arvon $\mathbf{x}_1$, X_2 saa arvon $\mathbf{x}_2$ jne. Jakaumia $p_{X_i}(\mathbf{x}_i)$ kutsutaan (osamuuttujan) X_i *marginaalijakauman tiheysfunktioksi*. Nämä saadaan kokonaisjakauman tiheysfunktiosta integroimalla muut muuttujat siitä pois. Marginaalijakaumien tiheysfunktioille pätevät kaikki tiheysfunktioiden ominaisuudet.

Satunnaismuuttujat $X_1, X_2, \ldots, X_n$ ovat *riippumattomia* jos ja vain jos

$$p_X(\mathbf{x}) = p_{(X_1,\ldots,X_n)}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = p_{X_1}(\mathbf{x}_1) \cdots p_{X_n}(\mathbf{x}_n). \tag{3.8}$$

Ajatellaan satunnaismuuttujiin X_1 ja X_2 liittyviä tapahtumia, vaikkapa E_i $i = 1, \ldots, n$ ja $F_j, j = 1, \ldots, m$. Tällöin satunnaismuuttujien tilastollisesta riippumattomuudesta seuraa että mielivaltaiset niihin liittyvät tapahtumat ovat tilastollisesti riippumattomia, ts.

$$P(E_i \cap F_j) = P(E_i)P(F_j),$$

mielivaltaisilla indeksien i ja j arvoilla. Satunnaismuuttujien tilastollinen riippumattomuus voidaankin tulkita niin, että niiden mallintamien (osa)kokeiden (mittausten) tulokset eivät riipu toisistaan. Usein satunnaismuuttujien tilastollinen riippumattomuus määritelläänkin kuten yllä. Määritelmä on ekvivalentti tiheysfunktion avulla tehtävän määritelmän kanssa.

Satunnaismuuttujiin voidaan luonnollisesti liittää myös ehdollisen todennäköisyyden käsite. Oletetaan, että meillä on kaksi satunnaismuuttujaa X ja Y ja, että satunnaismuuttujaan Y liittyvässä kokeessa realisoituu arvo $\mathbf{y}$. Tällöin saamme uuden satunnaismuuttujan, joka mallintaa X:n todennäköisyyttä olettaen, että $Y = \mathbf{y}$. Tällä satunnaismuuttujalla on luonnollisesti myös tiheysfunktio $p_{X|Y}$, jonka määritelmä perustuu X:n ja Y:n yhteisjakauman tiheysfunktioon $p_{X,Y}$ sekä Y:n marginaalijakauman tiheysfunktioon:

$$p_{X|Y}(\mathbf{x}|\mathbf{y}) = \frac{p_{X,Y}(\mathbf{x},\mathbf{y})}{p_Y(\mathbf{y})}. \tag{3.9}$$

Tiheysfunktion $p_{X|Y}$ määritelmä voidaan johtaa ehdollisen todennäköisyyden määritelmästä (3.5). Johto on helppo diskreetissä tapauksessa: Oletetaan, että $p_Y(\mathbf{y}) > 0$. Nyt

$$\begin{array}{rcll} p_{X|Y}(\mathbf{x}|\mathbf{y}) & = & P(X = \mathbf{x}|Y = \mathbf{y}) & \text{(tiheysfunktion määritelmä)} \\ & = & \frac{P(X=\mathbf{x},Y=\mathbf{y})}{P(Y=\mathbf{y})} & \text{(ehdollisen tod. näk. määritelmä)} \\ & = & \frac{p_{X,Y}(\mathbf{x},\mathbf{y})}{p_Y(\mathbf{y})}. & \text{(tiheysfunktion määritelmä)} \end{array}$$

Jatkuville tiheysfunktioille johto on kuitenkin työläs johtuen siitä, että tapahtuman $Y = \mathbf{y}$ todennäköisyys on aina nolla, joten jätämme sen tällä kurssilla väliin.

Vielä voidaan todeta, että satunnaismuuttujat X ja Y ovat tilastollisesti riippumattomia jos ja vain jos

$$p_{X|Y}(\mathbf{x},\mathbf{y}) = p_X(\mathbf{x})$$

kaikilla $\mathbf{x}, \mathbf{y}$.

3.7 Bayesin sääntö

Seuraavaksi käsittelemme hahmontunnistuksessa tärkeää tilastollista tulosta, joka on saanut nimensä Thomas Bayesin (1702 - 1761) mukaan. Tästäkin lauseesta on olemassa versiot tapahtumille ja tiheysfunktioille, jotka molemmat esitämme. Johdamme tuloksen tapahtumille.

Tapahtumien E ja F ehdollisen todennäköisyyden määritelmästä saadaan suoraan kertomalla puolittain $P(F)$:lla

$$P(E \cap F) = P(F)P(E|F).$$

Tämä siis olettaen, että $P(F) > 0$. Jos myös $P(E) > 0$, niin

$$P(E \cap F) = P(E)P(F|E).$$

Yhdistämällä saadaan

$$P(F)P(E|F) = P(E \cap F) = P(E)P(F|E).$$

Edelleen

$$P(F|E) = \frac{P(F)P(E|F)}{P(E)}.$$

Tätä tulosta kutsutaan *Bayesin säännöksi*. Erilainen esitysmuoto sille saadaan kun muistetaan lauseen 1 kohta 5: Eli siis, jos tapahtumat $F_1, \ldots, F_n$ ovat toisensa poissulkevia ja otosavaruus $S = F_1 \cup \cdots \cup F_n$, kaikille k on voimassa

$$P(F_k|E) = \frac{P(F_k)P(E|F_k)}{P(E)} = \frac{P(F_k)P(E|F_k)}{\sum_{i=1}^{n} P(F_i)P(E|F_i)}.$$

Esitämme nämä tulokset ja vastaavat tulokset tiheysfunktioille vielä lauseen muodossa.

Lause 4 *Olkoot $E, F, F_1, \ldots, F_n$ S:n tapahtumia. Lisäksi $F_1, \ldots, F_n$ muodostavat S:n partition, $P(E) > 0$, $P(F) > 0$. Tällöin*

1. $P(F|E) = \frac{P(F)P(E|F)}{P(E)}$;
2. $P(F_k|E) = \frac{P(F_k)P(E|F_k)}{\sum_{i=1}^{n} P(F_i)P(E|F_i)}$.

Olkoot X ja Y samaan kokeeseen liittyviä satunnaismuuttujia. Tällöin

3. $p_{X|Y}(\mathbf{x}|\mathbf{y}) = \frac{p_X(\mathbf{x})p_{Y|X}(\mathbf{y}|\mathbf{x})}{p_Y(\mathbf{y})}$;
4. $p_{X|Y}(\mathbf{x}|\mathbf{y}) = \frac{p_X(\mathbf{x})p_{Y|X}(\mathbf{y}|\mathbf{x})}{\int p_X(\mathbf{x})p_{Y|X}(\mathbf{y}|\mathbf{x})d\mathbf{x}}$.

Huomaa, että tiheysfunktioille tulos pätee vaikka toinen satunnaismuuttujista olisi jatkuva ja toinen olisi diskreetti, kunhan vain integraali korvataan tarvittaessa summalla.

Esimerkki. Syöpätestin tiedetään olevan täsmällinen 95%:ssa tapauksia, tarkoittaen, että sen herkkyys on 95% (jos testiin tulevalla henkilöllä on kyseinen syöpä, testi on positiivinen todennäköisyydellä 0.95) ja että sen tarkkuus on myös 95% (jos testiin tulevalla henkilöllä ei ole kyseistä syöpää, testi on negatiivinen todennäköisyydellä 0.95). Lisäksi tiedetään, että kyseisen syövän esiintymistodennäköisyys on 2000 tapausta 100000:sta. Millä todennäköissydellä testiin tulevalla henkilöllä on kyseinen syöpä, jos testin tulos on positiivinen?

Olisi houkuttelevaa vastata kysymykseen 95% (ja kun tehtävä on ollut tenttikysymyksenä, moni opiskelija on päätynyt tähän vastaukseen). 95 % on kuitenkin todennäköisyys sille, että testi on positiivinen, **jos henkilöllä jo tiedetään olevan ko. syöpä**. Samoin 95 % on todennäköisyys sille, että testi on negatiivinen, **jos tiedetään, että henkilöllä ei ole kyseistä syöpää**. Nämä ovat ehdollisia todennäköisyyksiä. Me olemme kiinnostuneita myös ehdollisesta todennäköisyydestä:

haluamme tietää onko koehenkilöllä syöpä **kun tiedämme, että testin tulos on positiivinen**. Tämä on eri todennäköisyys kuin kaksi edellistä, kiinnostava tapatuma ja ehto, joka on jo toteutunut ovat vaihtaneet tässä paikkaa. Pystymme siis laskemaan kiinnostavan todennäköisyyden käyttämällä Bayesin kaavaa.

Notaation yksinkertaistamiseksi, merkitään symbolilla C (cancer) tapausta, että henkilöllä on syöpä, symbolilla H (healthy) tapausta, että henkiöllä ei ole syöpää, symbolilla T (true) tapausta, että testin tulos on positiivinen, ja symbolilla F (false) tapausta, että testin tulos on negatiivinen. Annetut todennäköisyydet ovat

$$P(T|C) = 0.95, P(F|H) = 0.95.$$

Todennäköisyyden ominaisuuksien perusteella (tapaukset F ja T ovat toistensa komplementteja)

$$P(F|C) = 1 - P(T|C) = 0.05, P(T|H) = 1 - P(F|H) = 0.05.$$

Huomaa myös, että

$$P(C) = 0.02, P(H) = 0.98.$$

Bayesin kaavasta saadaan (tapaukset F ja T ovat toistensa komplementteja ja niin ollen muodostavat otosavaruuden partition)

$$P(C|T) = \frac{P(T|C)P(C)}{P(T|C)P(C) + P(T|H)P(H)} = \frac{0.95 \cdot 0.02}{0.95 \cdot 0.02 + 0.05 \cdot 0.98} = 0.2794,$$

joka on kysytty todennäköisyys.

3.8 Odotusarvo ja varianssi

Viittaamme tällä(kin) kurssilla joskus käsitteisiin odotusarvo ja varianssi, joten on syytä määritellä nämä käsitteet, vaikkeivat ne kurssin kannalta ole kovin keskeisiä. Jatkuvan satunnaismuuttujan X odotusarvo $E[X]$ määritellään seuraavasti:

$$E[X] = \int_{-\infty}^{\infty} \mathbf{x} p_X(\mathbf{x}) d\mathbf{x}.$$

Diskreetin satunnaismuuttujan tapauksessa integraali korvautuu summalla

$$E[X] = \sum_{\mathbf{x} \in D_X} \mathbf{x} p_X(\mathbf{x}),$$

missä D_X on X:n arvojoukko. Huomaa että d-komponenttia sisältävän satunnaismuuttujan odotusarvo on d komponenttinen vektori. Konkreettisesti odotusarvo on satunnaiskokeen odotettu lopputulema (keskimäärin).

Jatkuvan satunnaismuuttujan X varianssi määritellään

$$Var[X] = \int_{-\infty}^{\infty} (\mathbf{x} - E[X])(\mathbf{x} - E[X])^T p_X(\mathbf{x}) d\mathbf{x}.$$

Diskreetin satunnaismuuttujan tapauksessa integraali korvautuu summalla

$$Var[X] = \sum_{\mathbf{x} \in D_X} (\mathbf{x} - E[X])(\mathbf{x} - E[X])^T p_X(\mathbf{x}),$$

missä D_X on X:n arvojoukko. X:n varianssi on siis $d \times d$ matriisi. Erityisesti, jos $X \sim N(\mu, \Sigma)$, niin $E[X] = \mu$ ja $Var[X] = \Sigma$. Konkreettisesti varianssi kuvaa satunnaismuuttujan saamien arvojen hajontaa. Lisää esimerkkejä löytyy esimerkiksi osoitteesta `http://www.pori.tut.fi/~frank/toden/Odotusarvo.pdf`.

3.9 Moniulotteinen normaalijakauma

Moniulotteista normaalijakaumaa tarkastellaan tälläkin kurssilla usein, joten tutustumme siihen nyt tarkemmin. Ensin kuitenkin tarvitsemme positiivi-definiitin matriisin määritelmän: Symmetrinen $d \times d$ matriisi A on positiivi-definiitti, jos kaikille nollavektorista eroaville $\mathbf{x} \in \mathbb{R}^d$ on voimassa, että

$$\mathbf{x}^T A \mathbf{x} > 0.$$

Jotta ylläoleva epäyhtälö olisi järkevä, täytyy *neliömuodon* $\mathbf{x}^T A \mathbf{x} = \sum_{i=1}^d \sum_{j=1}^d a_{ij} x_i x_j$ olla skalaari. Tämä tarkoittaa, että vektorin $\mathbf{x}$ täytyy olla pystyvektori. Positiivi-definiitti matriisi A on ei-singulaarinen, joten on olemassa kääneteismatriisi A^{-1}, joka on myös positiivi-definiitti. Tämän todistus säästetään harjoitustehtäväksi. Myöskin positiivi-definiitin matriisin A determinantti $\det(A)$ on aina positiivinen.

Esimerkki. Esimerkiksi matriisi

$$B = \begin{bmatrix} 2 & 0.2 & 0.1 \\ 0.2 & 3 & 0.5 \\ 0.1 & 0.5 & 4 \end{bmatrix}$$

on positiivi-definiitti. Havainnollistetaan vielä neliömuodon laskemista valitsemalla $\mathbf{x} = \begin{bmatrix} -2 & 1 & 3 \end{bmatrix}^T$. Tällöin

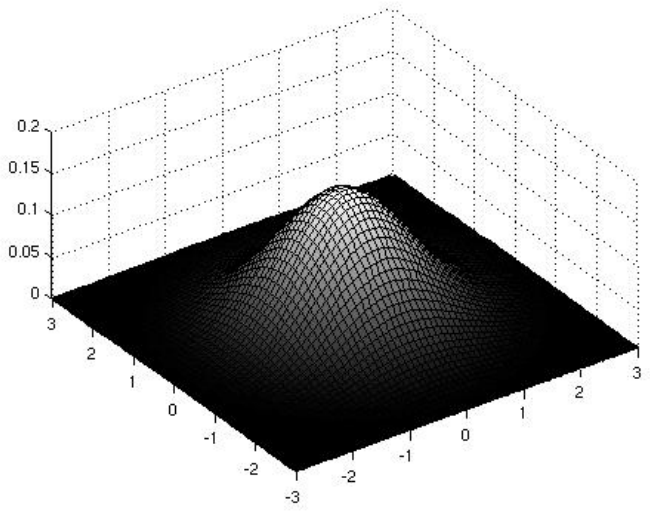
$$\begin{aligned} \mathbf{x}^T B \mathbf{x} &= 2 \cdot (-2) \cdot (-2) + 0.2 \cdot 1 \cdot (-2) + 0.1 \cdot 3 \cdot (-2) \\ &+ 0.2 \cdot (-2) \cdot 1 + 3 \cdot 1 \cdot 1 + 0.5 \cdot 3 \cdot 1 \\ &+ 0.1 \cdot (-2) \cdot 3 + 0.5 \cdot 1 \cdot 3 + 4 \cdot 3 \cdot 3 \\ &= 48. \end{aligned}$$

Normaalijakautuneella d-komponenttisella satunnaismuuttujalla $X \sim N(\mu, \Sigma)$ on tiheysfunktio

$$p_X(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}\sqrt{\det(\Sigma)}} \exp[-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)], \tag{3.10}$$

missä parametri μ d-komponenttinen vektori ja Σ $d \times d$ positiivi-definiitti matriisi. Parametrit μ ja Σ määrittävät X:n odotusarvon ja kovarianssin. Koska Σ^{-1} positiivi-definiitin matriisin käänteismatriisina on positiivi-definiitti $\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)$ on

positiivinen kaikilla $\mathbf{x}$ ja nolla ainoastaan kun $\mathbf{x} = \mu$. Tästä syystä tiheysfunktion arvo on ylhäältä rajoitettu. Koska $\det(\Sigma) > 0$, niin $p_X(\mathbf{x}) > 0$ kaikilla $\mathbf{x}$. Voidaan myös todeta, että $\int_{-\infty}^{\infty} p_X(\mathbf{x})d\mathbf{x} = 1$ (1-ulotteisen tapauksen todistus löytyy osoitteesta `http://mathworld.wolfram.com/GaussianIntegral.html`), joten (3.10) on todella tiheysfuktio.



Kuva 3.3: Kaksiulotteisen normaalijakauman tiheysfunktio

Yksi-ulotteisessa tapauksessa tiheysfunktioksi tulee

$$p_X(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp[-\frac{1}{2}(\frac{x-\mu}{\sigma})^2].$$

Esimerkki. Havainollistetaan vielä tiheysfunktion arvon laskua pisteessä $\mathbf{x}$ äskeisen neliömuotoesimerkin jatkoksi. Oletetaan, että $\mathbf{x} = [-1, 2, 4]^T, \mu = [1, 1, 1]^T$ ja

$$\Sigma = \frac{1}{23.33} \begin{bmatrix} 11.75 & -0.75 & -0.20 \\ -0.75 & 7.99 & -0.98 \\ -0.20 & -0.98 & 5.96 \end{bmatrix}$$

joten (tähän tarvitaan Matlabia)

$$\Sigma^{-1} = \begin{bmatrix} 2 & 0.2 & 0.1 \\ 0.2 & 3 & 0.5 \\ 0.1 & 0.5 & 4 \end{bmatrix}$$

kuten matriisi B edellä. Nyt $\mathbf{x} - \mu = [-2, 1, 3]^T$ ja edellisen neliömuotoesimerkin perusteella

$$s = (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) = 48$$

sekä

$$\exp(-0.5s) \approx 3.77 \cdot 10^{-11}.$$

Vielä kun havaitaan, että $\det(\Sigma) = 1/23.33$ (taas Matlab), niin tiheysfunktion arvoksi saadaan

$$p_X([-1, 2, 4]^T) \approx (\sqrt{23.33}/15.7496) \cdot 3.77 \cdot 10^{-11} = 1.16 \cdot 10^{-11},$$

eli todella pieni arvo. Vielä pari huomiota liittyen esimerkkiin. 1) Muista, että tiheysfunktion arvo yhdessä tietyssä pisteessä ei ole todennäköisyys millekkään tapahtumalle. Tälläisten todennäköisyystiheyksien laskeminen on kuitenkin yleistä hahmontunnistuksessa kuten seuraavassa luvussa käy ilmi. 2) Kaikki tämänkaltaiset laskut tehdään todellisuudessa tietokoneella numeerisesti laskien, käsin laskut ovat tässä vain esimerkin vuoksi, toivottavasti ne helpottavat monimuuttujatodennäköisyyksiin liittyvien laskutoimitusten ymmärtämistä.

Normaalijakaumalla on useita ominaisuuksia, jotka takaavat sille erityisaseman kaikkien jatkuvien satunnaismuuttujien jakaumien joukossa. Näistä muutama on hyvä tuntea tätäkin kurssia ajatellen. Merkitsemme jatkossa μ:n komponentteja μ_i ja matriisin Σ komponentteja σ_{ij}. Seuraavassa $X \sim N(\mu, \Sigma)$.

- X:n osamuuttujat $X_1, \ldots, X_d$ ovat normaalijakautuneita $X_i \sim N(\mu_i, \sigma_{ii})$.
- X:n osamuuttujat X_i ja X_j ovat riippumattomia jos ja vain jos $\sigma_{ij} = 0$.
- Olkoon $X \sim N(\mu, \Sigma)$ ja A ja B ei singulaarisia $d \times d$ matriiseja. Tällöin $AX \sim N(A\mu, A\Sigma A^T)$ ja AX ja BX ovat tilastollisesti riippumattomia jos ja vain jos $A\Sigma B^T$ on nollamatriisi.
- Kahden normaalijakautuneen satunnaismuuttujan summa on normaalijakautunut.

Todetaan vielä, että keskeisen raja-arvolauseen mukaan riippumattomien samoinjakautuneiden satunnaismuuttujien summa on normaalisti jakautunut kunhan muuttujien määrä lähestyy ääretöntä. Käytännössä ääretön ei tässä yhteydessä tarkoita kovin suurta lukua. Myöskin samoinjakautuneisuus voidaan korvata väljemmillä kriteereillä. Tämä osaltaan selventää miksi on hyödyllistä olettaa satunnaismuuttuja normaalijakautuneeksi siinä tapauksessa, että tätä vastaan ei ole perusteita.

Luku 4

Bayesin päätösteoria ja optimaaliset luokittimet

Bayesin päätösteoria muodostaa luokituksen teorian selkärangan. Se kertoo miten suunnitella paras mahdollinen luokitin kun hahmontunnistusongelman kaikki tilastolliset piirteet ovat tunnetut. Teoria on suhteellisen itsestäänselvien asioiden formalisointi ja se muodostaa vankan pohjan luokittimien myöhemmälle tarkastelulle.

Huomautus notaatiosta. Jatkossa olemme hieman huolettomampia notaation kanssa. Emme esimerkiksi indeksoi satunnaismuuttujan tiheysfunktiota satunnaismuuttujan symbolilla.

4.1 Luokitusongelma

Piirrevektorit $\mathbf{x}$, jotka tehtävänämme on luokitella, ovat *piirreavaruuden* vektoreita. Piirreavaruus on useimmiten d-ulotteinen Euklidinen avaruus $\mathbb{R}^d$, joskin se voi olla esimerkiksi myös binäärilukujonojen muodostama avaruus $\{0,1\}^d$. Yleisesti merkitsemme piirreavaruutta symbolilla $\mathbb{F}$.

Tehtävänämme on sijoittaa mielivaltainen piirrevektori $\mathbf{x} \in \mathbb{F}$ johonkin annetuista c:sta luokasta[1]. Luokkia merkitsemme $\omega_1, \ldots, \omega_c$. Huomaa, että tilastollisesti ajatellen tutkimme satunnaismuuttujaparia (X, ω), jossa X mallintaa piirrevektorin arvoa ja ω sen luokkaa.

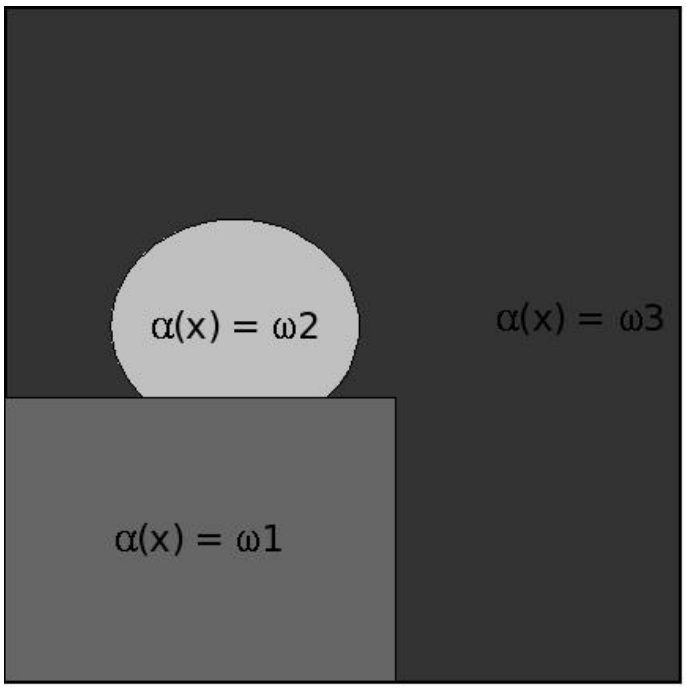
Luokista tunnetaan niiden *prioritodennäköisyydet* $P(\omega_1), \ldots, P(\omega_c)$ ja niiden *luokkatiheysfunktiot* $p(\mathbf{x}|\omega_1), \ldots, p(\mathbf{x}|\omega_c)$. Prioritodennäköisyys $P(\omega_i)$ määrittää kuinka suuri osa kaikista piirrevektoreista alun alkaen kuuluu luokkaan ω_i. Luokkatiheysfunktio $p(\mathbf{x}|\omega_i)$, niinkuin merkintä kertoo, määrittää luokaan ω_i kuuluvien piirrevektorien tiheyden avaruudessa $\mathbb{F}$. Tämä on sama asia kuin X:n tiheys ehdolla että sen luokka on ω_i. Täsmällisemmin nämä oletukset voi esittää sanomalla, että ω:n marginaalijakauma tunnetaan, ja että X:n jakauma ehdolla ω tunnetaan. Yleiset

[1] Itseasiassa tehtävänä on sijoittaa kohde, jonka abstrakti esitys piirrevektori on, johonkin luokkaan, joten sanonta ei ole täysin korrekti. Tämä epäkorrektius ei useinkaan ymmärtämystä haittaa ja se on hyvin tyypillinen luokituskirjallisuudelle, joten emme anna sen häiritä enempää. Joissain tapauksissa kuitenkin sanonta saattaa johtaa väärinkäsityksiin ja silloin muotoilemme sanamme tarkemmin

todennäköisyysjakaumia koskevat oletukset ovat voimassa: Esimerkiksi

$$\sum_{i=1}^{c} P(\omega_i) = 1.$$

Koska luokitusongelma oli sijoittaa mielivaltainen piirrevektori $\mathbf{x} \in \mathbb{F}$ johonkin annetuista c:sta luokasta, luokitin on itseasiassa kuvaus piirreavaruudelta luokkien joukkoon. Tätä kuvausta merkitsemme symbolilla α ja luokitustulos piirrevektorin ollessa $\mathbf{x}$ on $\alpha(\mathbf{x})$. Kuvausta α kutsutaan usein *päätössäännöksi*.



Kuva 4.1: Päätösalueet

Toinen tulkinta luokittimelle saadaan kun ajatellaan piirreavaruuden partitiota jonka se muodostaa: Jokainen piirreavaruuden vektori nimittäin kuuluu yksikäsitteiseen luokkaan ja näin ollen piirreavaruus voidaan jakaa osiin luokkien kesken (Kts. kuva 4.1). Merkitsemme näitä osia $\mathcal{R}_1, \ldots, \mathcal{R}_c$ ja kutsumme niitä *päätösalueiksi*. Selvästi päätössääntöön α liittyvät päätösalueet

$$\mathcal{R}_i = \{\mathbf{x} : \alpha(\mathbf{x}) = \omega_i\}.$$

Jos d on suuri, tällainen päätösalue-esitys luokittimelle ei luonnollisestikaan ole laskennallisesti järkevä, mutta sen tarjoamasta intuitiosta on usein hyötyä.

4.2 Luokitusvirhe

Koska luokitusongelma on luonteeltaan tilastollinen, emme voi olettaa että mikään luokitin asettaisi kaikkia kohteita oikeaan luokkaan: Voihan eri luokkaan kuuluvilla kohteilla olla sama piirrevektori. Tästä syystä on mielekästä tarkastella luokitusvirhettä: Todennäköisyyttä $E(\alpha)$, että kohde asetetaan väärään luokkaan. Koska

päätössääntö α liittää kohteeseen luokan ainoastaan sen piirrevektorin perusteella, riittää tarkastella todennäköisyyttä $E(\alpha(\mathbf{x})|\mathbf{x})$, että piirrevektorilla $\mathbf{x}$ varustetut kohteet (joita meidän ei siis ole mahdollista erottaa) luokittuvat väärin. Todennäköisyyksien perusominaisuuksista seuraa

$$E(\alpha(\mathbf{x})|\mathbf{x}) = 1 - P(\alpha(\mathbf{x})|\mathbf{x}), \tag{4.1}$$

missä siis $P(\alpha(\mathbf{x})|\mathbf{x})$ on todennäköisyys, että luokka $\alpha(\mathbf{x})$ on oikea. Luokittimen α luokitusvirhe voidaan kirjoittaa

$$E(\alpha) = \int_{\mathbb{F}} E(\alpha(\mathbf{x})|\mathbf{x})p(\mathbf{x})d\mathbf{x} = \int_{\mathbb{F}} [1 - P(\alpha(\mathbf{x})|\mathbf{x})]p(\mathbf{x})d\mathbf{x}. \tag{4.2}$$

Mutta mistä saadaan $P(\alpha(\mathbf{x})|\mathbf{x})$? Bayesin säännön perusteella

$$P(\alpha(\mathbf{x})|\mathbf{x}) = \frac{p(\mathbf{x}|\alpha(\mathbf{x}))P(\alpha(\mathbf{x}))}{p(\mathbf{x})} \tag{4.3}$$

ja luokitusvirhe on siis

$$E(\alpha) = 1 - \int_{\mathbb{F}} p(\mathbf{x}|\alpha(\mathbf{x}))P(\alpha(\mathbf{x}))d\mathbf{x}. \tag{4.4}$$

Huomaamisen arvoista on, että

$$p(\mathbf{x}|\alpha(\mathbf{x}))P(\alpha(\mathbf{x})) = p(\mathbf{x}, \alpha(\mathbf{x})).$$

Luokittimen α luokitusvirhe on siis yhtä suuri kuin tapahtuman $\{(\mathbf{x}, \alpha(\mathbf{x})) : \mathbf{x} \in \mathbb{F}\}$ komplementin todennäköisyys. (Notaatio $\{(\mathbf{x}, \alpha(\mathbf{x})) : \mathbf{x} \in \mathbb{F}\}$ on ehkä vaikea mieltää. Se tarkoittaa, että piirrevektori $\mathbf{x}$ saa kaikki mahdolliset arvot ja luokitus $\alpha(\mathbf{x})$ ratkaistaan $\mathbf{x}$:n perusteella.) Joukon $\{(\mathbf{x}, \alpha(\mathbf{x})) : \mathbf{x} \in \mathbb{F}\}$ komplementti on

$$\{(\mathbf{x}, \omega) : \mathbf{x} \in \mathbb{F}, \omega \in \{\omega_1, \ldots, \omega_c\}, \omega \neq \alpha(\mathbf{x})\}.$$

Päätösalueiden avulla luokitusvirhe voidaan kirjoittaa muodossa

$$E(\alpha) = \sum_{i=1}^{c} \int_{\mathcal{R}_i} [1 - p(\mathbf{x}|\omega_i)P(\omega_i)]d\mathbf{x} = 1 - \sum_{i=1}^{c} \int_{\mathcal{R}_i} p(\mathbf{x}|\omega_i)P(\omega_i)d\mathbf{x}, \tag{4.5}$$

missä siis $\mathcal{R}_i, i = 1, \ldots, c$ ovat päätössääntöön α liittyvät päätösalueet.

4.3 Bayesin minimivirheluokitin

Tässä kappaleessa esittelemme *Bayesin minimivirheluokittimen*, joka kappaleessa 4.1 esitettyjen tilastollisten seikkojen ollessa tiedossa minimoi luokitusvirheen.

Bayesin minimivirheluokitin määritellään

$$\alpha_{Bayes}(\mathbf{x}) = \arg \max_{\omega_i, i=1,\ldots,c} P(\omega_i|\mathbf{x}). \tag{4.6}$$

Merkintä $\arg\max_{\mathbf{x}} f(\mathbf{x})$ tarkoittaa funktion f argumentin $\mathbf{x}$ arvoa, jolla funktio saa maksimiarvonsa. Esimerkiksi, jos $f(x) = -(x-1)^2$, niin $\arg\max_x f(x) = 1$ (ja $\max f(x) = 0$). *Bayesin minimivirheluokitin siis valitsee luokan, joka on todennäköisin kun havaittu piirrevektori on* $\mathbf{x}$. *Posterior todennäköisyys* $P(\omega_i|\mathbf{x})$ lasketaan Bayesin säännön perusteella. Kun vielä huomataan, että $p(\mathbf{x})$ ei riipu luokasta ja $p(\mathbf{x}) \geq 0$ kaikille $\mathbf{x}$, niin saadaan toinen esitys Bayesin minimivirheluokittimelle:

$$\alpha_{Bayes}(\mathbf{x}) = \arg \max_{\omega_i, i=1,\ldots,c} p(\mathbf{x}|\omega_i)P(\omega_i). \tag{4.7}$$

Jos kahdella tai useammalla luokalla on sama ehdollinen todennäköisyys kun $\mathbf{x}$ on annettu, voidaan valita kumpi tahansa näistä luokista. Usein kutsumme Bayesin minimivirheluokitinta lyhyesti *Bayes-luokittimeksi*. Käytännössä Bayes-luokitus piirrevektorille $\mathbf{x}$ lasketaan laskemalla $p(\mathbf{x}|\omega_i)P(\omega_i)$ jokaiselle luokalle $\omega_i, i = 1, \ldots, c$ ja sijoittamalla $\mathbf{x}$ luokkaan, jolle $p(\mathbf{x}|\omega_i)P(\omega_i)$ on suurin.

Määritelmänsä nojalla Bayes-luokitin minimoi ehdollisen virheen $E(\alpha(\mathbf{x})|\mathbf{x}) = 1 - P(\alpha(\mathbf{x})|\mathbf{x})$ kaikille $\mathbf{x}$. Tästä ja integraalien perusominaisuuksista taas seuraa, että se minimoi myös luokitusvirheen. Tarkemmin sama voidaan ilmaista:

Lause 5 *Olkoon* $\alpha : \mathbb{F} \to \{\omega_1, \ldots, \omega_c\}$ *mielivaltainen päätössääntö (luokitin) ja* $\alpha_{Bayes} : \mathbb{F} \to \{\omega_1, \ldots, \omega_c\}$ *Bayes luokitin. Tällöin*

$$E(\alpha_{Bayes}) \leq E(\alpha).$$

Bayesin luokittimen luokitusvirhettä $E(\alpha_{Bayes})$ kutsutaan *Bayesin virheeksi* ja se on pienin mahdollinen luokitusvirhe kiinnitetylle luokitusongelmalle. Bayesin virhe voidaan kirjoittaa

$$E(\alpha_{Bayes}) = 1 - \int_{\mathbb{F}} \max_{\omega_i} [p(\mathbf{x}|\omega_i)P(\omega_i)] d\mathbf{x} \tag{4.8}$$

Lopuksi vielä oikaistaan yksi etenkin sovelluspuolella usein esiintyvä väärinkäsitys. Bayesin luokitin ei sisällä oletusta luokkatiheyksien normaalijakautuneisuudesta, vaan luokkatiheydet voivat olla mielivaltaisia tiheysfunktioita.

4.4 Bayesin minimiriskiluokitin

Bayesin minimivirheluokitin on erikoistapaus *Bayesin minimiriskiluokittimesta*. Oletetaan luvussa 4.1 mainittujen oletusten lisäksi, että on annettu a toimenpidettä (engl. action) $\alpha_1, \ldots, \alpha_a$. Nämä kuvaavat luokituksen seurauksena suoritettavaa toimenpidettä. Esimerkiksi pullonpalautusautomaatti luokittettuaan pullot antaa asiakkaalle kuitin yhteenlasketuista pullopanteista. Toimenpiteet liitetään luokitukseen *tappiofunktion* (engl. loss function) λ avulla. Sen arvo $\lambda(\alpha_i|\omega_j)$ kuvaa toimenpiteestä α_i aiheutuvaa tappiota kun todellinen luokka on ω_j. Päätössäännöt ovat nyt kuvauksia α piirreavaruudesta toimenpiteiden joukolle. Määrittelemme vielä *ehdollisen tappion* eli *ehdollisen riskin*

$$R(\alpha_i|\mathbf{x}) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j)P(\omega_j|\mathbf{x}), \tag{4.9}$$

joka toimenpiteestä α_i seuraa kun havaittu piirrevektori on $\mathbf{x}$. Bayesin minimiriskiluokitin yksinkertaisesti laskee ehdolliset riskit jokaiselle toimenpiteelle ja valitsee toimenpiteen, jonka ehdollinen riski on pienin.

Samaan tapaan kuin Bayes-luokitin myös Bayesin minimiriskiluokitin takaa optimaalisen luokitustuloksen. Se nimittäin minimoi kokonaisriskin

$$R_{total}(\alpha) = \int R(\alpha(\mathbf{x})|\mathbf{x})p(\mathbf{x})d\mathbf{x}, \tag{4.10}$$

päätössääntöjen α suhteen.

Bayes-luokitin saadaan minimiriskiluokittimen erikoistapauksena kun valitaan toimenpiteiksi α_i luokitus luokkaan ω_i ja tappiofunktioksi nolla-yksi tappiofunktio

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & \text{jos} \quad i = j \\ 1 & \text{jos} \quad i \neq j \end{cases}.$$

Toimenpiteiden määrä a voi olla eri suuri kuin c. Tästä on hyötyä esimerkiksi kun haluamme, että hahmontunnistusjärjestelmä voi erotella piirrevektorit joita ei voida riittävällä varmuudella luokittaa.

Esimerkki. Valaisemme Bayes-luokituksen ja minimiriskiluokituksen eroja vielä roskapostin luokitteluesimerkillä. Tuleva sähköpostiviesti kuuluu joko normaalipostiluokkaan ω_1 tai roskapostiluokkaan ω_2. On kaksi toimenpidettä α_1 (pidä sähköpostiviesti inboxissa) ja α_2 (viesti menee /dev/null:n). Koskapa normaalin viestin menettäminen on (yleensä) noin kolme kertaa tuskallisempaa kuin roskapostin saaminen inboxiin, valitsemme tappiofunktion

$$\begin{array}{ll} \lambda(\alpha_1|\omega_1) = 0 & \lambda(\alpha_1|\omega_2) = 1 \\ \lambda(\alpha_2|\omega_1) = 3 & \lambda(\alpha_2|\omega_2) = 0 \end{array}.$$

Lisäksi meille on ylhäältä annettu $P(\omega_1) = 0.4, P(\omega_2) = 0.6$ ja olemme viestin piirrevektorista laskeneet $p(\mathbf{x}|\omega_1) = 0.35, p(\mathbf{x}|\omega_2) = 0.65$.

Laskemme ensin posterior todennäköisyydet kummallekin luokalle:

$$P(\omega_1|\mathbf{x}) = \frac{0.35 \cdot 0.4}{0.35 \cdot 0.4 + 0.65 \cdot 0.6} = 0.264;$$
$$P(\omega_2|\mathbf{x}) = 0.736.$$

Bayesin minimiriskiluokitusta varten laskemme lisäksi ehdollisen riskin kummallekin toimenpiteelle. Ne ovat

$$R(\alpha_1|\mathbf{x}) = 0 \cdot 0.264 + 0.736 = 0.736,$$
$$R(\alpha_2|\mathbf{x}) = 0 \cdot 0.736 + 3 \cdot 0.264 = 0.792.$$

Eli siis Bayes-luokitin on sitä mieltä, että viesti on roskapostia. Minimiriskiluokitin silti antaa sen sujahtaa inboxiin, koska virheluokituksen tappio on tässä tapauksessa pienempi.

Huomautettakoon, että loppumonisteessa toimenpiteet, tappiot ja riskit unohdetaan ja keskitytään ainoastaan luokitusvirheeseen luokituksen onnistumista kuvaavana indikaattorina. Useat esiteltävistä käytännön luokittimista kuitenkin mukautuvat yleisempään viitekehykseen helposti.

4.5 Erotinfunktiot ja päätöspinnat

Bayes-luokitinta käsitellessämme huomasimme, että sama luokitin voidaan esittää useammalla eri tavalla. Yleisesti luokitin määritellään usein apufunktioiden, ns. *erotinfunktioiden* avulla. Jokaisella luokalla ω_i on oma erotinfunktionsa $g_i(\mathbf{x})$, joka ottaa syötteekseen piirrevektorin $\mathbf{x}$. Luokitin asettaa siten piirrevektorin $\mathbf{x}$ luokkaan ω_i, jos

$$g_i(\mathbf{x}) > g_j(\mathbf{x})$$

kaikille $i \neq j$.

Bayes-luokittimelle erotinfunktioksi voidaan valita

$$g_i(\mathbf{x}) = P(\omega_i|\mathbf{x}), i = 1, \ldots, c$$

tai

$$g_i(\mathbf{x}) = p(\mathbf{x}|\omega_i)P(\omega_i), i = 1, \ldots, c.$$

Erotinfunktiot eivät siis ole yksikäsitteisiä, ja tietty luokitin voidaan esittää useiden erilaisten erotinfunktioiden avulla. Toisaalta tietyt erotinfunktiot määräävät yksikäsitteisen luokittimen. Seuraava tulos voidaan todistaa helposti:

Lause 6 *Olkoon $f : \mathbb{R} \to \mathbb{R}$ monotonisesti kasvava, ts. $f(x) < f(y)$ aina kun $x < y$. Tällöin erotinfunktiot*

$$g_i(\mathbf{x}), i = 1, \ldots, c$$

ja

$$f(g_i(\mathbf{x})), i = 1, \ldots, c$$

määrittelevät saman luokittimen.

Siispä erotinfunktioihin voidaan lisätä vakioita, kertoa niitä positiivisilla vakioilla tai niistä voidaan ottaa logaritmi mitenkään muuttamatta niiden esittämää luokitinta.

Myöhemmin olemme kiinnostuneita erityisesti muotoa

$$g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_{i0}$$

olevista erotinfunktioista. Näitä kutsutaan *lineaarisiksi erotinfunktioiksi*. Yllä $\mathbf{w}_i$ on samaa dimensionaalisuutta piirrevektorien kanssa ja w_{i0} on skalaari. Luokitinta, joka voidaan esittää yksinomaan lineaaristen erotinfunktioiden avulla kutsutaan *lineaariseksi luokittimeksi*.

Kun luokkia on vain kaksi, luokitin on kätevää esittää yhden ja ainoan erotinfunktion avulla:

$$g(\mathbf{x}) = g_1(\mathbf{x}) - g_2(\mathbf{x}). \tag{4.11}$$

Tällöin, jos $g(\mathbf{x}) > 0$, niin $\mathbf{x}$ sijoitetaan luokkaan ω_1, ja muutoin $\mathbf{x}$ sijoitetaan luokkaan ω_2.

Kuten jo aikaisemmin todettiin, luokitin jakaa piirreavaruuden päätösalueisiin luokkien kesken. Päätösalueet voidaan esittää erotinfunktioiden avulla näppärästi:

$$\mathcal{R}_i = \{\mathbf{x} : g_i(\mathbf{x}) > g_j(\mathbf{x}) \forall i \neq j\}.$$

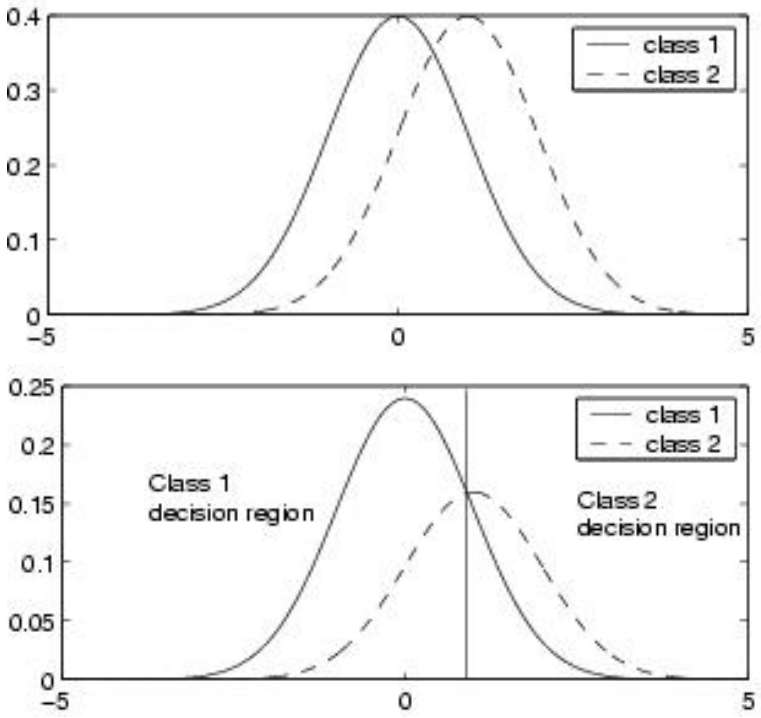
Päätösalueiden rajapintoja, ts. joukkoja

$$\mathcal{R}_{ij} = \{\mathbf{x} : g_i(\mathbf{x}) = g_j(\mathbf{x}), g_k(\mathbf{x}) < g_i(\mathbf{x}) \forall k \neq i, j\}.$$

kutsutaan *päätöspinnoiksi*. Niiden muodon perusteella luokittimen 'geometriaa', ts. miten luokitin jakaa päätösavaruuden osiin luokkien kesken on usein helpompi hahmotella kuin suoraan päätösalueiden perusteella. Voidaan esimerkiksi havaita, että lineaaristen luokittimien yksittäiset päätöspinnat ovat janoja piirreavaruudessa.

Esimerkki. Ajatellaan kahden luokan luokitusongelmaa, jossa $P(\omega_1) = 0.6$, $P(\omega_2) = 0.4$, $p(x|\omega_1) = \frac{1}{\sqrt{2\pi}} \exp[-0.5x^2]$ ja $p(x|\omega_2) = \frac{1}{\sqrt{2\pi}} \exp[-0.5(x-1)^2]$. Tehtävänä on etsiä päätösalueet Bayes-luokittimelle.

Päätösalue $\mathcal{R}_1$ on niiden piirrevektoreiden x joukko joille $P(\omega_1|x) > P(\omega_2|x)$. Päätösalue $\mathcal{R}_2$ on niiden piirrevektoreiden x joukko joille $P(\omega_2|x) > P(\omega_1|x)$. Päätöspinta on niiden piirrevektoreiden x joukko joille $P(\omega_2|x) = P(\omega_1|x)$.



Kuva 4.2: Yläpaneli: luokkatiheydet. Alapaneli: $P(\omega_1|x)$, $P(\omega_2|x)$ ja päätösalueet.

Aloitetaan päätöspinnan määrittämisellä:

$$P(\omega_1|x) = P(\omega_2|x) \Rightarrow p(x|\omega_1)P(\omega_1) = p(x|\omega_2)P(\omega_2),$$

missä käytettiin Bayesin sääntöä ja kerrottiin $p(x)$:llä. Edelleen

$$\begin{aligned} & p(x|\omega_1)P(\omega_1) = p(x|\omega_2)P(\omega_2) \\ \Rightarrow \quad & \ln[p(x|\omega_1)P(\omega_1)] = \ln[p(x|\omega_2)P(\omega_2)] \\ \Rightarrow \quad & -(x^2/2) + \ln 0.6 = -(x-1)^2/2 + \ln 0.4 \\ \Rightarrow \quad & x^2 - 2\ln 0.6 = x^2 - 2x + 1 - 2\ln 0.4. \end{aligned}$$

Päätöspinta, piste tässä tapauksessa, on $x^* = 0.5 + \ln 0.6 - \ln 0.4 \approx 0.91$. Päätösalueet ovat (katso kuva 4.2) $\mathcal{R}_1 = \{x : x < x^*\}$, $\mathcal{R}_2 = \{x : x > x^*\}$.

4.6 Erotinfunktiot normaalijakautuneille luokille

Moniulotteinen normaalijakauma on suosituin malli luokkatiheysfunktioille. Tämä perustuu kahteen seikkaan: Normaalijakauma on matemaattisesti mukava käsitellä. Ja se tarjoaa hyvän mallin tilanteessa, joissa kutakin luokkaa edustavien piirrevektorien voidaan ajatella olevan kohinaisia versioita luokalle tyypillisestä piirrevektorista. Tässä kappaleessa tarkastelemme Bayesin luokittimen erotinfunktioita, päätöspintoja ja päätösalueita kun luokkatiheyksiä mallinnetaan normaalijakauman avulla. Tämän kappaleen on tarkoitus demonstroida esiteltyjä käsitteitä, ja siksipä tulosten johdot ovat tärkeämpiä kuin itse tulokset. Tosin eivät tuloksetkaan turhia ole.

d-ulotteisen normaalijakauman tiheysfunktio on

$$p_{normal}(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}\sqrt{\det(\Sigma)}} \exp[-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)], \tag{4.12}$$

missä jakauman parametrit ovat $\mu \in \mathbb{R}^d$ ja positiividefiniitti $d \times d$ matriisi Σ.

Nyt siis oletamme, että luokkatiheysfunktioita mallinnetaan normaalijakaumalla ja prioritodennäköisyydet voivat olla mielivaltaisia. Merkitsemme luokan tiheysfunktion parametreja μ_i, Σ_i, ts. $p(\mathbf{x}|\omega_i) = p_{normal}(\mathbf{x}|\mu_i, \Sigma_i)$.

Aloitamme kaavassa (4.7) määritellystä Bayes-luokittimesta, josta saamme erotinfunktiot

$$g_i(\mathbf{x}) = p_{normal}(\mathbf{x}|\mu_i, \Sigma_i)P(\omega_i). \tag{4.13}$$

Kun kaavan (4.13) oikeasta puolesta otetaan logaritmi (jonka saamme lauseen 6 nojalla tehdä) ja kirjoitetaan normaalijakauman tiheys auki saadaan[2]

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1}(\mathbf{x} - \mu_i) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln\det(\Sigma_i) + \ln P(\omega_i). \tag{4.14}$$

Erotamme kolme eri tapausta:

1. $\Sigma_i = \sigma^2 I$, missä σ^2 on (positiivinen) skalaari ja I on identiteettimatriisi.

2. $\Sigma_i = \Sigma$, ts. kaikilla luokilla on sama kovarianssimatriisi.

3. Σ_i on mielivaltainen.

[2]Tässä vaiheessa on hyvä huomauttaa, että kaavojen (4.13) ja (4.14) erotinfunktiot $g_i(\mathbf{x})$ eivät suinkaan ole samoja funktioita vaan ne toimivat yleisenä merkintänä (saman luokittimen) erotinfunktioille. Täsmällisempi notaatio kuitenkin aiheuttaisi enemmän päävaivaa kuin siitä olisi hyötyä.

Tapaus 1

Kaava (4.14) voidaan kirjoittaa

$$g_i(\mathbf{x}) = -\frac{||\mathbf{x} - \mu_i||^2}{2\sigma^2} + \ln P(\omega_i), \tag{4.15}$$

kun oikealta puolelta jätetään vakiotermit pois. Symbolilla $||\cdot||$ merkitään Euklidista normia, siis

$$||\mathbf{x} - \mu_i||^2 = (\mathbf{x} - \mu_i)^T(\mathbf{x} - \mu_i).$$

Kirjoittamalla normin auki saamme

$$g_i(\mathbf{x}) = -\frac{1}{2\sigma^2}(\mathbf{x}^T\mathbf{x} - 2\mu_i^T\mathbf{x} + \mu_i^T\mu_i) + \ln P(\omega_i). \tag{4.16}$$

Koska termi $\mathbf{x}^T\mathbf{x}$ on sama kaikille luokille, sekin voidaan jättää pois, jolloin

$$g_i(\mathbf{x}) = \frac{1}{\sigma^2}(\mu_i^T\mathbf{x} - \frac{1}{2}\mu_i^T\mu_i) + \ln P(\omega_i), \tag{4.17}$$

joka on lineaarinen erotinfunktio, jossa

$$\mathbf{w}_i = \frac{\mu_i}{\sigma^2},$$

ja

$$w_{i0} = \frac{-\mu_i^T\mu_i}{2\sigma^2} + P(\omega_i).$$

Luokittimen päätösalueita tapauksessa 1 on havainnollistettu kuvassa 4.3, joka on kuva 2.10 Dudan, Hartin ja Storkin kirjasta.

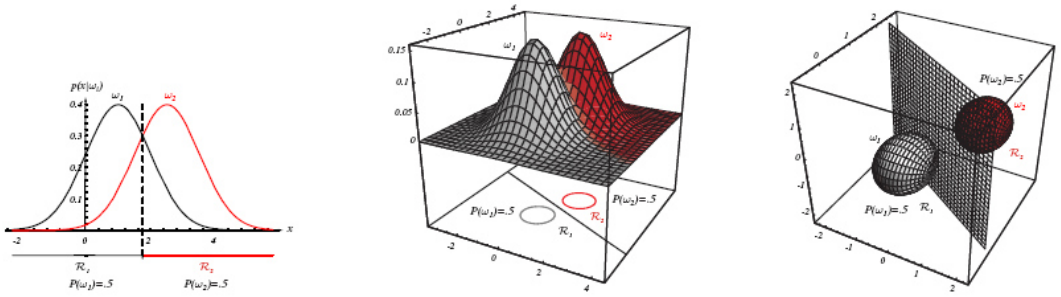


FIGURE 2.10. If the covariance matrices for two distributions are equal and proportional to the identity matrix, then the distributions are spherical in d dimensions, and the boundary is a generalized hyperplane of $d-1$ dimensions, perpendicular to the line separating the means. In these one-, two-, and three-dimensional examples, we indicate $p(\mathbf{x}|\omega_i)$ and the boundaries for the case $P(\omega_1) = P(\omega_2)$. In the three-dimensional case, the grid plane separates $\mathcal{R}_1$ from $\mathcal{R}_2$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Kuva 4.3: Tapaus 1

Tärkeä erikoistapaus erotinfunktioista (4.15) saadaan kun $P(\omega_i) = \frac{1}{c}$ kaikilla i. Tällöin piirrevektori $\mathbf{x}$ sijoitetaan siihen luokkaan, jonka keskiarvovektori on lähinnä $\mathbf{x}$:ää. Tämä on *pienimmän etäisyyden luokitin* ja se on lineaarisen luokittimen erikoistapauksena luonnollisesti myös lineaarinen.

Tapaus 2

Vaikka piirteet olisivatkin toisistaan riippuvia, mutta Σ_i:t ovat edelleen kaikille luokille samat (ts. $\Sigma_i = \Sigma$), luokitin on silti lineaarinen. Se voidaan esittää erotinfunktioiden

$$g_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} + w_{i0}, \tag{4.18}$$

missä

$$\mathbf{w}_i = \Sigma^{-1}\mu_i$$

ja

$$w_{i0} = -\frac{1}{2}\mu_i^T \Sigma^{-1}\mu_i + \ln P(\omega_i),$$

avulla.

Luokittimen päätösalueita tapauksessa 2 on havainnollistettu kuvassa 4.4, joka on kuva 2.11 Dudan, Hartin ja Storkin kirjasta.

Tapaus 3

Nyt emme oleta mitään ylimääräistä luokkatiheysfunktioista. Tässä tapauksessa erotinfunktioita

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1}(\mathbf{x} - \mu_i) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln\det(\Sigma_i) + \ln P(\omega_i)$$

ei voida paljoa sieventää: vain termi $\frac{d}{2}\ln 2\pi$ voidaan jättää pois. Tässä tapauksessa erotinfunktiot eivät enää ole lineaarisia, vaan *neliöllisiä*. Tämä johtaa huomattavasti monimutkaisempiin päätösalueisiin kuin kahdessa edellisessä tapauksessa. Nyt päätospinnat ovat yleisiä toisen asteen pintoja, eikä päätösalueiden enää tarvitse olla edes jatkuvia.

Luokittimen päätösalueita tapauksessa 3 on havainnollistettu kuvassa 4.5, joka on kuva 2.14 Dudan, Hartin ja Storkin kirjasta.

4.7 Riippumattomat binääripiirteet

Tarkastelemme vielä tapausta jossa piirteet ovat binääriarvoisia, siis kukin piirre saa joko arvon nolla tai yksi (tai vastaavasti 'ei' tai 'kyllä'). Koska piirteet ovat diskreettejä, piirrevektoria edustava satunnaismuuttuja on myös diskreetti ja siis luokkatiheysfunktiot ovat diskreetin muuttujan tiheysfunktioita. Luokituksen teoriaan tällä ei oikeastaan ole mitään vaikutusta, integraalit vain täytyy korvata summilla. Muuten kaikki vanhat tulokset ja kikat ovat käytettävissä.

Oletamme vielä, että piirteet ovat tilastollisesti riippumattomia ja kunkin piirteen luokkatiheydet noudattavat Bernoullin jakaumaa. Tästä saamme

$$P(\mathbf{x}|\omega_i) = \prod_{j=1}^{d} q_{ij}^{x_j}(1 - q_{ij})^{(1-x_j)}, \tag{4.19}$$

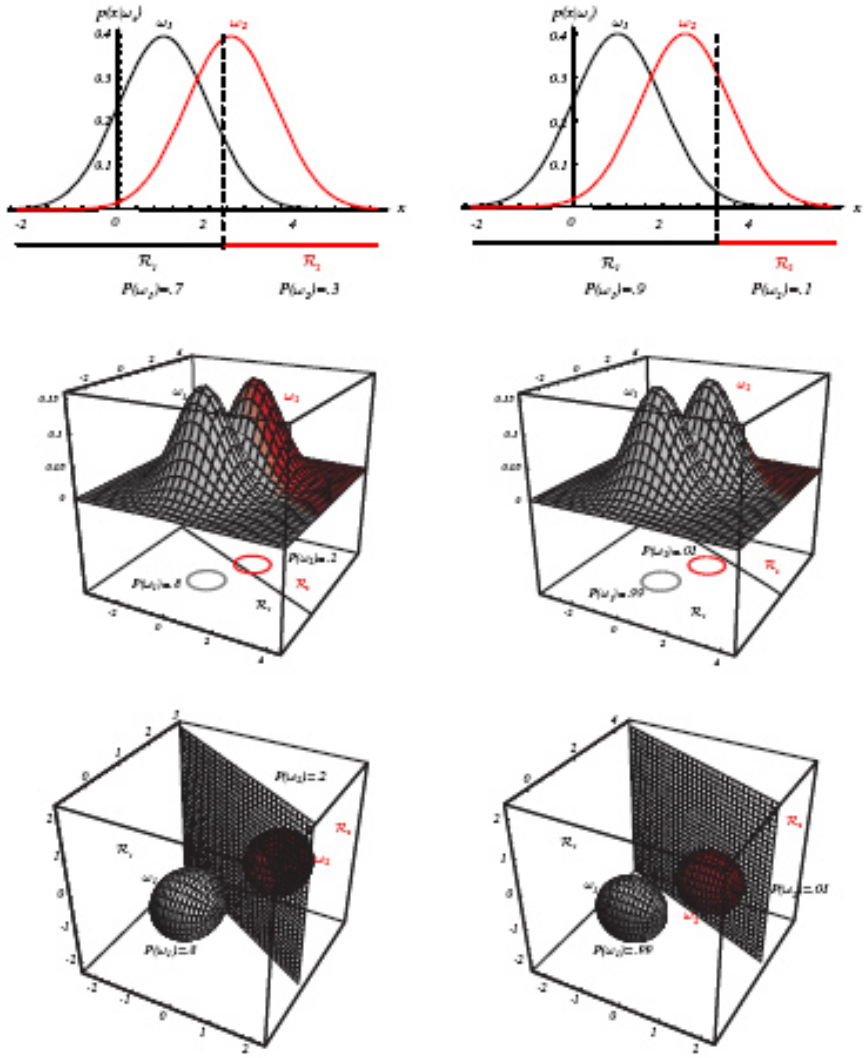


FIGURE 2.11. As the priors are changed, the decision boundary shifts; for sufficiently disparate priors the boundary will not lie between the means of these one-, two- and three-dimensional spherical Gaussian distributions. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Kuva 4.4: Tapaus 2

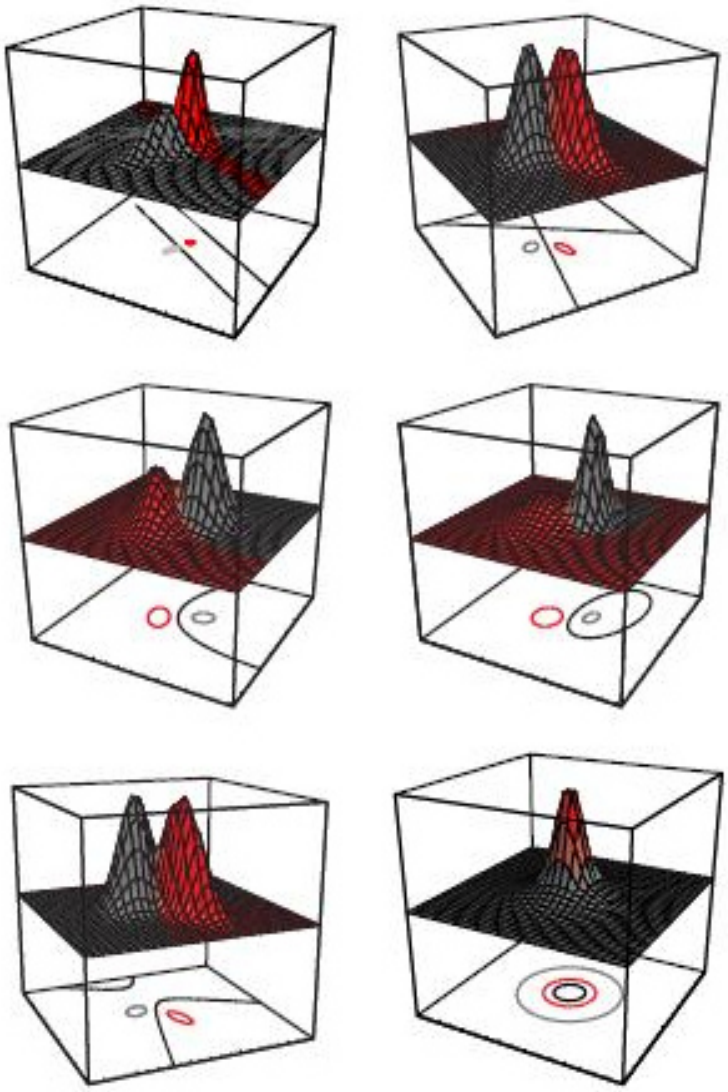


FIGURE 2.14. Arbitrary Gaussian distributions lead to Bayes decision boundaries that are general hyperquadrics. Conversely, given any hyperquadric, one can find two Gaussian distributions whose Bayes decision boundary is that hyperquadric. These variances are indicated by the contours of constant probability density. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Kuva 4.5: Tapaus 3

missä $q_{ij}, j = 1, \ldots, d$ ovat luokan ω_i tiheyden parametrit. Jakauman tulkinta on, että j:s piirre vastaa kyllä tai ei kysymykseen kyllä (siis 1) todennäköisyydellä q_{ij}. Korostettakoon vielä, että piirteet ovat tilastollisesti riippumattomia. Hahmontunnistuksen kannalta, jos $q_{1j} > q_{2j}$, niin j:s piirre saa arvon 1 useammin, jos sen esittämä kohde kuuluu luokkaan ω_1 kuin jos kohde kuuluisi luokkaan ω_2.

Bayes-luokittimelle

$$\alpha_{Bayes}(\mathbf{x}) = \arg \max_{\omega_i, i=1,\ldots,c} P(\omega_i|\mathbf{x}).$$

erotinfunktiot voidaan kirjoittaa Bayesin kaavan perusteella muodossa

$$g_i(\mathbf{x}) = P(\mathbf{x}|\omega_i)P(\omega_i).$$

Sijoittamalla tähän binomijakauman mukaiset luokkatiheysfunktiot saadaan

$$g_i(\mathbf{x}) = \left[\prod_{j=1}^{d} q_{ij}^{x_j}(1 - q_{ij})^{(1-x_j)}\right] P(\omega_i). \tag{4.20}$$

Ottamalla oikeasta puolesta logaritmi ja sieventämällä saadaan

$$g_i(\mathbf{x}) = \sum_{j=1}^{d} [x_j \ln q_{ij} + \ln(1 - q_{ij}) - x_j \ln(1 - q_{ij})] + \ln P(\omega_i). \tag{4.21}$$

Erotinfunktiot ovat lineaarisia

$$g_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} + w_{i0} = \sum_{j=1}^{d} w_{ij} x_j + w_{i0},$$

missä

$$w_{ij} = \ln \frac{q_{ij}}{1 - q_{ij}},$$

ja

$$w_{i0} = \sum_{j=1}^{d} \ln(1 - q_{ij}) + \ln P(\omega_j).$$

Kahden luokan tapauksessa ainoaksi erotinfunktioksi muodostuu

$$g(\mathbf{x}) = g_1(\mathbf{x}) - g_2(\mathbf{x}) = \sum_{j=1}^{d} \ln \frac{q_{1j}(1 - q_{2j})}{(1 - q_{1j})q_{2j}} x_j + \sum_{j=1}^{d} \ln \frac{(1 - q_{1j})}{(1 - q_{2j})} + \ln \frac{P(\omega_1)}{P(\omega_2)}.$$

Käytämme vielä hetken luokitusvirheen suuruuden koon tarkasteluun riippumattomien piirteiden määrän kasvaessa. Tämä on mukavampaa diskreetissä tapauksessa, koska laskutoimitukset ovat suoraviivaisia ja esimerkiksi integraalien numeeriseen evaluointiin ei tarvitse puuttua. Jotta suoraviivaiset simulaatiot olisivat mahdollisia, täytyy d:n olla kuitenkin kohtuullisen pieni. Seuraavassa teemme joukon tilannetta yksinkertaistavia oletuksia, jotta voimme kasvattaa d:tä. Tässä tulokset ovat tärkeämpiä kuin niiden johto, jonka esitämme tiiviissä muodossa.

Luokitusvirhe on

$$E(\alpha_{Bayes}) = 1 - \sum_{\mathbf{x}} \max P(\mathbf{x}|\omega_i)P(\omega_i), \tag{4.22}$$

jossa $\mathbf{x}$ käy läpi kaikki d:n pituiset binäärivektorit. Siis mitä suurempi on todennäköisimmän luokan todennäköisyys, sitä pienempi on luokitusvirhe. Tarkastellaan kahden luokan tapausta, jossa luokkatiheyksiä mallinnetaan binomijakaumalla, ts. $q_{ij} = q_i$ kaikille j. Luokkatiheydet ovat

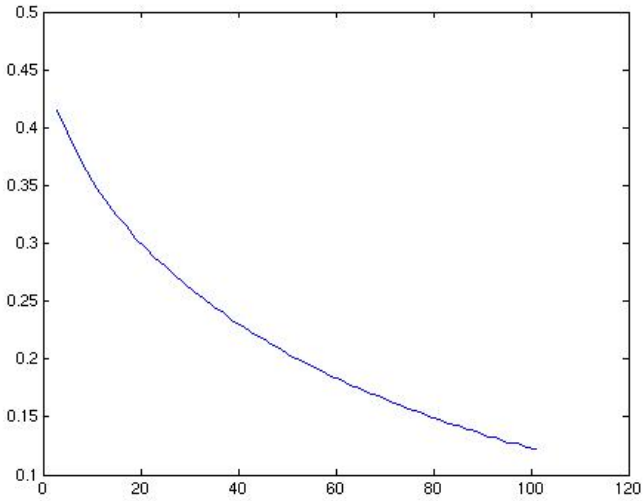
$$P(\mathbf{x}|\omega_i) = P(\sum x_j = k|q_i) = q_i^k(1-q_i)^{d-k}, i = 1, 2,$$

missä k on ykkösten lukumäärä piirrevektorissa $\mathbf{x}$. Oletetaan vielä, että luokkien prioritodennäköisyydet ovat yhtä kuin 0.5 ja että $q_1 > q_2$. Selkeästi $\mathbf{x}$:n luokan ratkaisee kuinka monta ykköstä $\mathbf{x}$:ssä on. Bayes-luokitin on lineaarinen ja luokittaa $\mathbf{x}$:n luokkaan ω_1 kun ykkösten määrä ylittää tietyn kynnysarvon t. Kynnysarvo voidaan laskea kaavan (4.21) perusteella. Luokitusvirhe on

$$\begin{aligned} E(\alpha_{Bayes}) &= 1 - \sum_{\mathbf{x}:k\le t} 0.5q_2^k(1-q_2)^{d-k} + \sum_{\mathbf{x}:k>t} 0.5q_1^k(1-q_1)^{d-k} \\ &= 1 - \sum_{k=0}^{t} 0.5\frac{d!}{k!(d-k)!}q_2^k(1-q_2)^{d-k} + \sum_{k=t+1}^{d} 0.5\frac{d!}{k!(d-k)!}q_1^k(1-q_1)^{d-k}. \end{aligned}$$

Huomaa, että luokitusvirheen kaavassa summataan nyt kaikkien k:den eikä kaikkien piirrevektorien $\mathbf{x}$ yli.

Luokitusvirheen suuruus d:n suhteen tapauksessa $q_1 = 0.3, q_2 = 0.2$ on esitetty kuvassa 4.6. Huomattavaa on että luokitusvirhe pienenee d:n kasvaessa. Tämä on yleisemminkin totta Bayes-luokittimelle. Tämä ei kuitenkaan tarkoita, että muiden, ei-optimaalisten, luokitinten virhe vähenisi piirteiden määrän kasvaessa. Esimerkiksi, jos äskeisten oletusten ollessa voimassa olisimme jostain syystä valinneet $t = \lfloor d/2 \rfloor$ (joka ei siis ole Bayes-luokitin), saisimme luokitusvirheen, joka hiljalleen kasvaisi kohti arvoa 0.5! (Merkintä $\lfloor d/2 \rfloor$ tarkoittaa suurinta kokonaislukua, joka on pienempi tai yhtä suuri kuin $d/2$.)



Kuva 4.6: Luokitusvirhe kun luokkatiheydet ovat binomijakauman mukaisia

Luku 5

Ohjattu luokkatiheysfunktioiden oppiminen

5.1 Ohjattu oppiminen ja Bayes luokitin

Edellisessä luvussa oletimme, että luokkatiheysfunktiot $p(\mathbf{x}|\omega_i)$ ja prioritodennäköisyydet $P(\omega_i)$ olivat tiedossa. Näin ei käytännössä ole asian laita, vaan nämä täytyy opetella. Tässä luvussa tarkastelemme, kuinka nämä on mahdollista oppia opetusnäytteiden perusteella. Tarkastelemme nyt siis nimenomaisesti ohjattua oppimista.

Kuten jo luvussa 2 määriteltiin, opetusnäytteet ovat piirrevektoreita, joiden korrektit luokat ovat tiedossa. Englanniksi näitä kutsutaan useilla eri nimillä, tyypillisimmin *training data* tai *training set*. Näitä opetus-piirrevektoreita on jokaisesta luokasta $\omega_1, \ldots, \omega_c$. Voimme järjestää nämä piirrevektorit luokittain ja merkitä luokkaan ω_i kuuluvien opetusnäytteiden joukkoa

$$\mathcal{D}_i = \{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i}\},$$

missä n_i on luokkaan ω_i kuuluvien opetusnäytteiden lukumäärä. Oletamme, että kunkin $\mathcal{D}_i$:n näytteet ovat realisaatioita tilastollisesti riippumattomista satunnaismuuttujista (siis eri kohteista), joiden kaikkien voidaan olettaa lisäksi olevan jakautuneita tiheysfunktion $p(\mathbf{x}|\omega_i)$ mukaisesti. Tällaista näytejoukkoa kutsutaan englanniksi *independent and identically distributed* eli lyhyesti i.i.d.

Opetusdata voi olla kerätty kahdella toisistaan eroavalla tavalla. Näillä eroilla on merkitystä prioritodennäköisyyksien oppimisessa. *Sekoiteotannassa* on sattumanvaraisesti valittu kohteet, laskettu niiden piirrevektorit ja sitten luokitettu kohteet sopivimpaan luokkaan. *Erillisessä otannassa* puolestaan on erikseen hankittu opetusnäytteet kustakin luokasta. Hahmontunnistusjärjestelmien opetuksen kannalta otantatekniikoiden erona on se, että sekoiteotannan näytteiden perusteella voidaan päätellä prioritodennäköisyydet $P(\omega_1), \ldots, P(\omega_c)$:

$$P(\omega_i) = \frac{n_i}{\sum_{j=1}^{c} n_j}. \tag{5.1}$$

Erillisen otannan perusteella ei sen sijaan voida luotettavasti päätellä prioritodennäköisyyksiä ja ne on viisainta olettaa tunnetuiksi.

Luokkatiheysfunktioiden kohdalla tilanne luonnollisesti on monimutkaisempi, tarvitseehan nyt opetella ääretön määrä arvoja $p(\mathbf{x}|\omega_i)$ äärellisen opetusnäytemäärän perusteella. Seuraavissa kappaleissa tutkimme miten luokkatiheysfunktioita voidaan estimoida opetusdatan perusteella.

5.2 Tiheysfunktion parametrien estimointi

5.2.1 Parametrien estimoinnin idea

Ensin tarkastelemme tilannetta, jossa tiheysfunktioiden $p(\mathbf{x}|\omega_i)$ estimointi yksinkertaistetaan parametrivektorin θ_i estimoinniksi olettamalla tiheysfunktioille tunnettu parametrinen muoto. Oletamme siis, että $p(\mathbf{x}|\omega_i)$ on tietyn jakaumaperheen jäsen ja toistaiseksi tuntematon parametrivektori θ_i määrittää mikä näistä luokkatiheys itseasiassa on. Esimerkiksi joissain tilanteissa on perusteltua olettaa, että luokat ovat normaalijakautuneita, jossa tapauksessa tarvittavat parametrit ovat μ_i, Σ_i kullekin luokalle. Perustelut voivat nojautua esimerkiksi (piirrevektoreiden) mittausjärjestelmän fysikaalisiin ominaisuuksiin.

Merkitsemme

$$p(\mathbf{x}|\omega_i) = p(\mathbf{x}|\omega_i, \theta_i),$$

missä voimme tulkita, että ω_i määrittää oletetun jakaumaperheen ja parametrivektori määrittää tuon perheen jäsenen. Korostettakoon, että jakaumaperheiden ei tarvitse olla samoja kaikille luokille. Esimerkiksi luokka ω_1 voi olla normaalijakautunut kun taas luokan ω_2 noudattaessa Cauchyn jakaumaa. Koska opetusnäytteet ovat riippumattomia, niin parametrivektorit voidaan estimoida kullekin luokalle erikseen. Tehtävänä nyt siis estimoida parametrivektorin θ_i arvoa opetusnäytteiden

$$\mathcal{D}_i = \{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i}\},$$

perusteella.

5.2.2 Suurimman uskottavuuden estimaatti

Merkintöjä yksinkertaistaaksemme unohdamme hetkeksi luokkia kuvaavat indeksit. Siis ongelmana on estimoida parametrivektoria θ opetusdatan $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ perusteella. Opetusdatan oletetaan olevan i.i.d. ja jakautuneen tiheysfunktion $p(\mathbf{x}|\theta)$ mukaisesti.

Suurimman uskottavuuden estimaatti (engl. maximum likelihood estimate) $\hat{\theta}$ maksimoi opetusdatan todennäköisyyden parametrivektorin θ suhteen. $\mathcal{D}$:n todennäköisyys voidaan kirjoittaa

$$p(\mathcal{D}|\theta) = \prod_{i=1}^{n} p(\mathbf{x}_i|\theta).$$

Joten

$$\hat{\theta} = \arg\max \prod_{i=1}^{n} p(\mathbf{x}_i|\theta). \tag{5.2}$$

Funktiota $p(\mathcal{D}|\theta)$ kutsutaan tässä yhteydessä *uskottavuusfunktioksi* (engl. likelihood function). Usein käytännössä on kätevämpää tarkastella uskottavuusfunktion sijaan sen logaritmia. Koska logaritmi on monotonisesti kasvava funktio uskottavuusfunktion logaritmin maksimointi johtaa täsmälleen samaan tulokseen kuin itse uskottavuusfunktion maksimointi. Niinpä voimme kirjoittaa

$$\hat{\theta} = \arg\max \sum_{i=1}^{n} \ln p(\mathbf{x}_i|\theta). \tag{5.3}$$

Käytämme suurimman uskottavuuden estimaatista jatkossa vierasperäistä lyhennettä *ML-estimaatti*.

5.2.3 ML-estimaatin löytäminen

ML-estimaatissa on se mukava puoli, että se voidaan tarvittaessa löytää, tai ainakin sitä voidaan etsiä numeerisesti. Joissakin tärkeissä tapauksissa ML-estimaatti pystytään määräämään analyyttisesti. Tämä tapahtuu seuraavan proseduurin avulla:

1. Etsi uskottavuusfunktion tai sen logaritmin gradientti.
2. Etsi gradientin nollakohdat ja kaikki muut kriittiset pisteet. (Kriittisiä pisteitä ovat gradientin nollakohtien lisäksi mm. maksimoitavan funktion epäjatkuvuuskohdat).
3. Evaluoi uskottavuusfunktio tai sen logaritmi kaikissa kriittisissä pisteissä ja valitse maksimiarvon antava kriittinen piste ML-estimaatiksi.

Vielä varoituksen sana: ML-estimaattia ei välttämättä ole kaikille jakaumille olemassa. Esimerkiksi uskottavuusfunktio saattaa kasvaa rajatta. Yksinkertaisille parametrisille malleille kuitenkin ML estimaatit ovat olemassa.

5.2.4 ML-estimaatit normaalijakaumalle

Johdamme nyt ML-estimaatit tapauksessa, jossa luokkatiheydet ovat normaalijakautuneita, ts.

$$p(\mathbf{x}|\theta) = \frac{1}{(2\pi)^{d/2}\sqrt{\det(\Sigma)}} \exp[-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)].$$

Ensin keskiarvovektori, siis $\theta = \mu$, ja kovarianssimatriisin Σ oletamme tunnetuksi. Uskottavuusfunktion logaritmi

$$l(\theta) = \sum_{i=1}^{n} \ln p(\mathbf{x}_i|\mu) = -\sum_{i=1}^{n} 0.5(\ln[(2\pi)^{d/2}\det(\Sigma)] + (\mathbf{x}_i-\mu)^T\Sigma^{-1}(\mathbf{x}_i-\mu)).$$

Sen gradientti

$$\nabla l(\mu) = -\sum_{i=1}^{n} \Sigma^{-1}(\mathbf{x}_i-\mu).$$

Asetetaan $\nabla l(\mu) = \mathbf{0}$. Koska Σ on ei-singulaarinen, niin Σ^{-1} on ei-singulaarinen ja gradientin nollakohdalla on yksikäsitteinen ratkaisu. Tämä todetaan helposti myös lokaaliksi maksimiksi, ja niin ollen nollakohta

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i, \tag{5.4}$$

on myös ML-estimaatti. Eli siis tässä tapauksessa ML-estimaatti on sama kuin $\mathcal{D}$:n *näytekeskiarvo*. Johdossa ei oikeastaan tarvittu tietoa kovarianssimatriisista vaan joistakin sen ominaisuuksista, joten tulos on voimassa vaikka sekä μ että Σ olisivat tuntemattomia.

Kovarianssimatriisin tapauksessa johto on hieman mutkikkaampi, joten jätämme sen väliin. Tuloksena saamme:

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T, \tag{5.5}$$

jossa $\hat{\mu}$ on kaavassa (5.4) määritelty parametrin μ ML-estimatti.

Vielä numeerinen esimerkki:

```
 x_1  x_2   x_3    x_4   x_5
 13  -115   119    33   -19
 29   119    -4    17    73

mu =  6.2000
      46.8000

sigma =  5762.6   -3212.2
        -3212.2    1937.0
```

5.2.5 ML-estimaatin ominaisuuksia

Parametrien estimointimenetelmiä on useita ja siksi onkin tärkeää tietää perusasiat ML-estimaattien ominaisuuksista. ML-estimaatilla, sen lisäksi että se voidaan numeerisesti ratkaista, on muutamia houkuttelevia teoreettisia ominaisuuksia. Seuraavassa niistä lyhyesti:

ML-estimaattorien voidaan yleisesti todistaa olevan *asymptoottisesti harhattomia*. Kansan kielellä ne antavat keskimäärin oikean tuloksen kunhan vain näytteitä on mielivaltaisen paljon. Tarkemmin parametrivektorin ML-estimaatin odotusarvo yli kaikkien n:n kokoisten näytejoukkojen on yhtä suuri kuin parametrivektorin todellinen arvo kun n lähestyy ääretöntä. Yleisesti ML-estimaatit eivät ole *harhattomia*, ts. jos n on äärellinen yllämainittu tulos ei välttämättä päde. Esimerkiksi normaalijakauman kovarianssimatriisin ML-estimaatti ei ole harhaton. Tämän lisäksi

kun n lähestyy ääretöntä, ML estimaatin voidaan todistaa olevan paras harhaton estimaattori siinä mielessä että sen varianssi on pienin mahdollinen. Vielä tämän lisäksi ML-estimaatit ovat *asymptoottisesti konsistentteja*, mikä tarkoittaa että ML-estimaatti on todennäköisyydellä 1 mielivaltaisen lähellä oikeaa parametrin arvoa kunhan n lähestyy ääretöntä.

Lähes kaikki ML-estimaattien optimaalisuusominaisuudet ovat niin sanottuja suuren näytekoon ominaisuuksia. Jos n on pieni, ML estimaattien yleisistä ominaisuuksista voidaan sanoa erittäin vähän.

5.2.6 Luokitusesimerkki

Tarkastelemme vielä ML-estimaattiin perustuvaa luokitinta Fisherin iirisdataan perustuvan esimerkin avulla. Otamme opetusnäytteiksi kymmenen ensimmäistä mittausta kustakin lajikkeesta. Oletamme luokkien olevan normaalijakautuneita ja että prioritodennäköisyydet ovat kaikille kolmelle luokalle $\frac{1}{3}$. Voimme esittää kaikkien luokkien harjoitusjoukot matriisimuodossa, jossa kukin vaakarivi vastaa yhtä opetusnäytettä. Luokalle ω_1 (iiris setosa) opetusnäytteet ovat:

```
    5.1000    3.5000    1.4000    0.2000
    4.9000    3.0000    1.4000    0.2000
    4.7000    3.2000    1.3000    0.2000
    4.6000    3.1000    1.5000    0.2000
    5.0000    3.6000    1.4000    0.2000
    5.4000    3.9000    1.7000    0.4000
    4.6000    3.4000    1.4000    0.3000
    5.0000    3.4000    1.5000    0.2000
    4.4000    2.9000    1.4000    0.2000
    4.9000    3.1000    1.5000    0.1000
```

Näistä saamme

```
mu1 = 4.8600    3.3100    1.4500    0.2200

Sigma1 =
    0.0764    0.0634    0.0170    0.0078
    0.0634    0.0849    0.0155    0.0148
    0.0170    0.0155    0.0105    0.0040
    0.0078    0.0148    0.0040    0.0056
```

Samoin muille luokille:

```
mu2 = 6.1000    2.8700    4.3700    1.3800
Sigma2 =
    0.4760    0.1740    0.2910    0.0720
    0.1740    0.1041    0.1191    0.0384
    0.2910    0.1191    0.2141    0.0584
```

```
    0.0720    0.0384    0.0584    0.0256

mu3 = 6.5700    2.9400    5.7700    2.0400
Sigma3 =
    0.5821    0.1252    0.4141    0.0782
    0.1252    0.1024    0.1052    0.0794
    0.4141    0.1052    0.3241    0.0752
    0.0782    0.0794    0.0752    0.0764
```

Ja nyt luokitin onkin sitten kasassa. Voimme nyt luokittaa piirrevektorin $\mathbf{x} = [\ 5.9 \quad 4.2 \quad 3.0 \quad 1.5\ ]^T$ laskemalla erotinfunktioiden (4.13) arvot kaikille luokille. Luokitin on sitä mieltä, että tässä on kyseessä luokka ω_2 eli iiris versicolor.

5.3 Tiheysfunktion epäparametrinen estimointi

Usein hahmontunnistuksessa tulee eteen tilanteita, jolloin luokkatiheysfunktioille ei ole mielekästä olettaa jotain tiettyä parametrista muotoa. Edellisessä kappaleessa kuvattu parametrien estimointi ei tällöin ole mahdollista. Sen sijaan meidän täytyy turvautua tiheysfunktioiden epäparametriseen estimointiin. Edellisen kappaleen ML-menetelmään verrattuna yritämme nyt löytää estimaatin tiheysfunktioille $p(\mathbf{x}|\omega_i)$ jokaisessa pisteessä $\mathbf{x}$ sen sijaan, että yrittäisimme estimoida tuntemattoman parametrivektorin arvoa. Voimme myös yrittää löytää estimaatin posterior luokkatodennäköisyyksille $P(\omega_i|\mathbf{x})$ suoraan olettaen, että opetusdata on asianmukaisesti (sekoiteotannalla) generoitu.

Tiheyden estimoinnin perusongelma formuloidaan seuraavasti: Oletetaan, että on annettu opetusdata $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ jossa näytteet ovat i.i.d. jakautuneet tuntemattoman tiheysfunktion $p(\mathbf{x})$ mukaan. Tehtävänä on löytää estimaatti $\hat{p}(\mathbf{x})$ tiheysfunktiolle $p(\mathbf{x})$ jokaisessa pisteessä $\mathbf{x}$.

Käsittelemme seuraavissa kappaleissa kahta erityyppistä estimointimenetelmää, Parzen ikkunoita ja k:n lähimmän naapurin menetelmää, ja niiden sovellusta luokitusongelmiin. Mutta ensin esittelemme histogrammit ja käsittelemme tiheyden estimointia yleisemmällä tasolla.

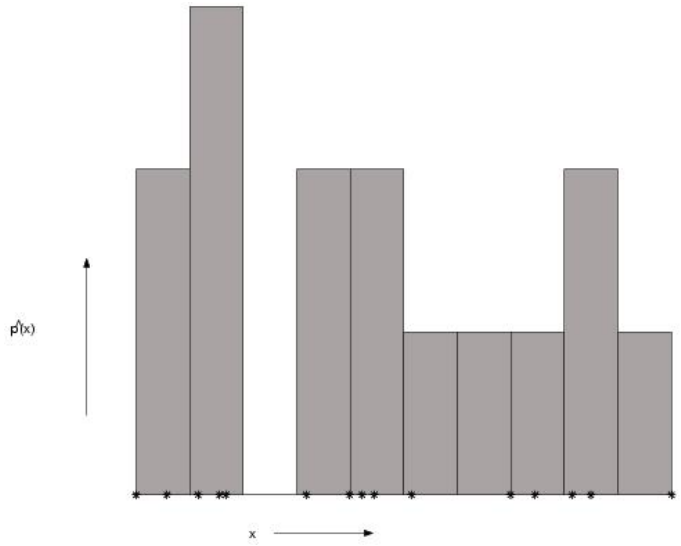
5.3.1 Histogrammi

Histogrammit ovat suoraviivaisin tapa estimoida tuntematonta tiheysfunktiota. Piirreavaruus jaetaan m samankokoiseen 'lokeroon' $\mathcal{B}_i$ (kts. Kuva 5.1) ja lasketaan kuinka monta opetusvektoria kuhunkin lokeroon osuu. Tätä merkitsemme symbolilla n_i. Tiheysestimaatti on nyt

$$\hat{p}(\mathbf{x}) = \frac{n_i}{Vn}, \quad \text{kun} \quad \mathbf{x} \in \mathcal{B}_\text{i}, \tag{5.6}$$

missä V on lokeroiden tilavuus.

Histogrammit - näin kapeasti määriteltynä - eivät ole kovinkaan hyviä tiheysestimaatteja, näin erityisesti d-ulotteisten jatkuva-arvoisten piirteiden tapauksessa.



Kuva 5.1: Histogrammi

Ensinnäkin estimaatit ovat epäjatkuvia, toisekseen sopivaa lokeronkokoa on vaikea arvioida.

5.3.2 Tiheyden estimoinnin yleinen formulointi

Yleistetään histogrammia sallimalla mielivaltaisen muotoiset ja kokoiset lokerot ja tarkastellaan mistä syystä oikein päädyimme niihin. Tarkastellaan nyt estimaattia $\hat{p}(\mathbf{x})$ pisteessä $\mathbf{x}$ ja aluetta $\mathcal{B}$ $\mathbf{x}$:n ympärillä. Todennäköisyys, että tietty opetusnäyte $\mathbf{x}_j$ osuu alueelle $\mathcal{B}$ on

$$P = \int_{\mathcal{B}} p(\mathbf{x}) d\mathbf{x}. \tag{5.7}$$

Tarvitsemme vielä todennäköisyyden sille, että k kappaletta n:stä opetusnäytteestä osuu alueelle $\mathcal{B}$. Koska näytteet ovat riippumattomia ja kukin näyte osuu alueelle $\mathcal{B}$ todennäköisyydellä P, voimme tarkastella tilannetta binomitodennäköisyyden avulla. Siis: k näytettä osuu alueelle todennäköisyydellä

$$P_k = \left(\begin{array}{c} n \\ k \end{array} \right) P^k (1-P)^{n-k}, \tag{5.8}$$

missä

$$\left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n!}{k!(n-k)!}$$

on erilaisten k:n kokoisten $\mathcal{D}$:n osajoukkojen määrä.

k:n odotusarvo Binomilauseen nojalla on

$$E[k] = \sum_{k=0}^{n} k P_k = nP, \tag{5.9}$$

Nyt, jos $E[k]$:ta estimoidaan havaitulla $\mathcal{B}$:hen osuvien opetusnäytteiden määrällä $\hat{k}$, niin saamme P:lle estimaatin

$$\hat{P} = \hat{k}/n. \tag{5.10}$$

Edelleen, jos $p(\mathbf{x})$ on jatkuva ja $\mathcal{B}$ on riittävän pieni,

$$\int_{\mathcal{B}} p(\mathbf{x})d\mathbf{x} \simeq p(\mathbf{x})V,$$

missä V on $\mathcal{B}$:n tilavuus. Tästä saadaan estimaatti (tai paremminkin täsmällinen tulkinta sille)

$$\hat{p}(\mathbf{x}) = \frac{\hat{k}}{nV} \simeq \frac{\int_{\mathcal{B}} p(\mathbf{x})d\mathbf{x}}{\int_{\mathcal{B}} d\mathbf{x}}. \tag{5.11}$$

Tästä tulkinnasta voidaan tehdä seuraavat päätelmät:

- Näin saatu tiheysestimaatti on tilakeskiarvoistettu versio todellisesta tiheysfunktiosta. Estimaatti muuttuu tarkemmaksi V:n pienentyessä. Kuitenkin, jos n samaan aikaan säilyy samankokoisena kuin ennen, niin ennen pitkää alueesta $\mathcal{B}$ tulee liian pieni, jotta se sisältäisi yhtään näytettä. Tämä taas johtaa hyödyttömään tiheysestimaattiin.

- Yleisemmin voidaan sanoa että tiheyden estimoinnissa täytyy hyväksyä ja ottaa huomioon kaksi seikkaa. 1) Tiheysestimaatti on aina keskiarvoistettu versio todellisesta tiheysfunktiosta ja 2) estimaatti $\hat{k}$ on vain estimaatti nP:sta kun rajoitetun suuruinen opetusnäytejoukko on saatavilla.

Joten peruskysymys kuuluu kuinka valita $\mathcal{B}$ ja niin ollen V. Teoreettisesti voidaan ajatella, että meillä on rajoittamaton määrä opetusnäytteitä käytössämme. Tällöin voimme muodostaa sarjan lokeroita $\mathcal{B}_1, \mathcal{B}_2, \ldots$, jotka kukin sisältävät pisteen $\mathbf{x}$. Lokeroa $\mathcal{B}_1$ käytetään kun opetusnäytteitä on yksi, $\mathcal{B}_2$ on käytössä kun opetusnäytteitä on kaksi ja niin edelleen. Merkitsemme $\mathcal{B}_n$:n tilavuutta symbolilla V_n ja $\mathcal{B}_n$:aan osuvien näytteiden määrää symbolilla k_n. Näihin perustuva tiheysestimaatti on

$$p_n(\mathbf{x}) = \frac{k_n}{nV_n}.$$

Jos nyt haluamme että,

$$\lim_{n\to\infty} p_n(\mathbf{x}) = p(\mathbf{x}),$$

niin kolmen ehdon täytyy olla voimassa,

1. $\lim_{n\to\infty} V_n = 0$
2. $\lim_{n\to\infty} k_n = \infty$
3. $\lim_{n\to\infty} k_n/n = 0$.

Vaikka ehdot näyttävät vaikeilta tulkita, ne eivät itseasiassa ole sitä. Ensimmäinen ehto takaa että todellista tiheyttä ei keskiarvoisteta jos näytteitä on ääretön määrä. Toinen ehto takaa että k_n/n on riittävän hyvä estimaatti P:sta. (Huomaa, että tämä ehto on järkevä ainoastaan jos $p(\mathbf{x}) > 0$). Ehto kolme tarkoittaa, että vain häviävän pieni osa opetusnäytteistä voi kuulua alueeseen $\mathcal{B}_n$.

Miten sitten taata yllä olevien ehtojen täyttyminen? Mahdollisuuksia lähteä tutkimaan asiaa on kaksi. Joko kiinnitämme V_n:n ja estimoimme parametria k_n. Tai sitten kiinnitämme k_n:n ja estimoimme sen sijaan parametria V_n. Huomaa että kummallakaan tyylillä ei aluetta $\mathcal{B}_n$ tarvitse välttämättä varsinaisesti määrittää.

Tiheysestimointiin liittyvät optimaalisuustulokset ovat pakostakin asymptoottista muotoa. Nimittäin Rosenblatt todisti jo vuonna 1956, että äärelliseen näytekokoon perustuvat tiheysestimaatit (jatkuvassa tapauksessa) ovat pakostakin harhaisia.

5.4 Parzen ikkunat

5.4.1 Histogrammeista Parzen ikkunoihin

Johdantona Parzen ikkunoihin tarkastelemme kappaleen 5.3.2 histogrammin yleistystä uudella tavalla. Oletamme ensin, että alue $\mathcal{B}_n$ on d-dimensionaalinen hyperkuutio. Tämän tilavuuden oletamme olevan $V_n = h_n^d$. Määrittelemme vielä funktion, joka saa arvon 1 origokeskisen yksikköhyperkuution sisällä ja arvon 0 sen ulkopuolella

$$\varphi(\mathbf{u}) = \begin{cases} 1 & \text{jos} \quad |\mathrm{u_j}| \leq 1/2 \\ 0 & \text{muulloin} \end{cases} \tag{5.12}$$

Tällöin, jos $\mathbf{x}$ on $\mathcal{B}_n$:n keskikohdassa, ja jos opetusnäyte $\mathbf{x}_i$ sijaitsee myös hyperkuutiossa $\mathcal{B}_n$, niin $\varphi((\mathbf{x} - \mathbf{x}_i)/h_n) = 1$. Muulloin $\varphi((\mathbf{x} - \mathbf{x}_i)/h_n) = 0$. Opetusnäytteiden lukumäärä $\mathcal{B}_n$:ssä on tällöin

$$k_n = \sum_{i=1}^{n} \varphi(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}).$$

Tiheysestimaatiksi saadaan

$$p_n(\mathbf{x}) = \frac{k_n}{nV_n} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V_n} \varphi(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}) \tag{5.13}$$

Yllä oleva estimaatti eroaa kappaleessa 5.3.1 määritellystä histogrammista siinä, että lokeroita ei ole ennalta määrätty vaan ne generoidaan opetusdatan perusteella ja siinä, että lokerot voivat olla toistensa 'päällä'.

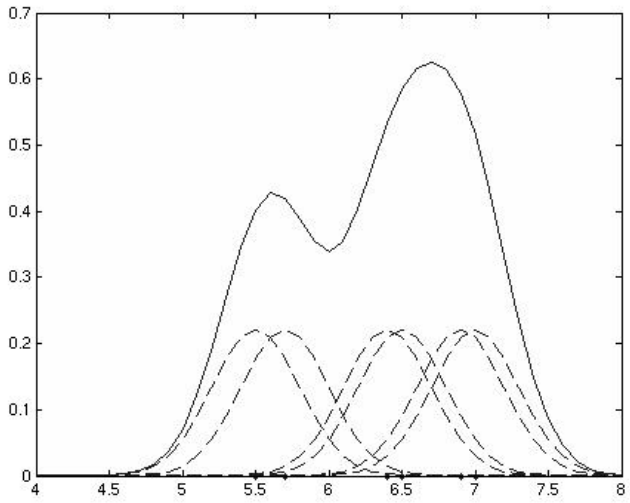
5.4.2 Tiheysfunktion Parzen-estimaatti

Tiheysfunktion *Parzen-estimaatti* saadaan kun yleistetään kaavan (5.13) estimaattia sallimalla muunkinlaiset φ funktiot kuin yllä, joita kutsutaan *ikkunafunktioksi*.

Tiheyden pisteessä $\mathbf{x}$, $p(\mathbf{x})$, Parzen estimaatti on

$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\varphi(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}). \tag{5.14}$$

Parzen-estimaatti pisteessä $\mathbf{x}$ on siis keskiarvo ikkunafunktion arvoista eri pisteissä (kts. Kuva 5.2). Tyypillisesti ikkunafunktiot ovat symmetrisiä tiheysfunktioita, joiden moodi (ts. maksimitiheys) on origossa. Normaalijakauman tiheysfunktio $\varphi(\mathbf{u}) = \frac{1}{(2\pi)^{d/2}}\exp[-0.5\mathbf{u}^T\mathbf{u}]$ on yleisin valinta ikkunafunktioksi. Tyypillisesti $V_n = h_n^d$, mutta huomaa että V_n:n geometrinen tulkinta hyperkuution tilavuutena ei ole enää täysin validi. Puutumme *ikkunanpituuden* h_n valintaan kappaleessa 5.4.4.



Kuva 5.2: Parzen estimaatti

Parzen-estimaatteja kutsutaan joissain yhteyksissä myös *ydinestimaateiksi* (engl. Kernel estimates). Parzen-estimaatit esitteli ensimmäisenä amerikkalainen tilastomatemaatikko Emanuel Parzen vuonna 1962, tosin jo Rosenblatt tutki samantyyppisiä estimaatteja vuonna 1956.

5.4.3 Parzen-estimaattien ominaisuuksia

Parzen-estimaateilla on monia mielenkiintoisia ominaisuuksia, joista tässä tutkimme muutamia. Ensin, jotta p_n olisi ehta tiheysfunktio, siltä vaaditaan (luvun 3 mukaan) 1) että $p_n(\mathbf{x}) \geq 0$ kaikilla $\mathbf{x}$ ja että 2) $\int p_n(\mathbf{x})d\mathbf{x} = 1$. Nämä ehdot täyttyvät kunhan vain ikkunafunktio on ehta tiheysfunktio, siis

1. $\varphi(\mathbf{x}) \geq 0$ kaikille $\mathbf{x}$
2. $\int \varphi(\mathbf{x})d\mathbf{x} = 1$.

Todistuksen säästämme harjoitustehtäväksi.

Jos puolestaan opetusnäytteiden määrä on ääretön, niin Parzen-estimaatit $p_n(\mathbf{x})$ tarkentuvat (esimerkiksi mean-square mielessä pisteittäin) todelliseen tiheyden arvoon kunhan seuraavat oletukset ovat voimassa:

1. p on jatkuva;
2. Ikkunafunktio on rajoitettu ja validi tiheysfunktio;
3. Ikkunafunktion arvot äärettömyydessä (kun argumenttivektorin pituus lähenee ääretöntä) ovat mielivaltaisen pieniä;
4. $V_n \to 0$ kun $n \to \infty$.
5. $nV_n \to \infty$ kun $n \to \infty$.

Oletuksia täytyy siis tehdä ikkunafunktion ja ikkunanleveyden valinnan lisäksi myös todellisesta tiheysfunktiosta. Ikkunafunktiota koskevat ehdot 2 ja 3 ovat hyvin luonnollisia ja esimerkiksi normaalijakauman tiheysfunktio on niiden mukainen.

Hahmontunnistuksen kannalta nämä ominaisuudet ovat tärkeitä koska näin voidaan todistaa, että Parzen-ikkunoihin perustuvan luokittimen virhe lähestyy Bayes-virhettä kunhan opetusnäytteiden määrä kasvaa kohti ääretöntä. Tämä pätee yleisemminkin: Kunhan tiheysestimaatit tarkentuvat tietyssä mielessä kohti todellisia luokkatiheyksiä ja prioritodennäköisyydet ovat oikein valittuja, niin tuloksena olevan luokittimen virhe lähestyy Bayes-virhettä. Emme tällä kurssilla määrittele tarkentumisehtoja tämän täsmällisemmin, koska näiden tarkka määrittely veisi turhan paljon aikaa.

5.4.4 Ikkunanleveyden valinta

Edellisen kappaleen vaatimukset täyttävä ikkunanleveys voidaan valita

$$h_n = h_1/\sqrt{n}.$$

Käytännössä täytyy kuitenkin vielä määrittää sopiva h_1. Tämä ei ole helppo tehtävä ja valinta onkin usein sovellusspesifinen ja perustuu siihen minkälaisia ominaisuuksia tiheysestimaateilta sovelluksen puolesta odotetaan. Jos h_1 on liian pieni tiheysestimaatista tulee liian epätasainen ja niin ollen se johtaa liian monimutkaiseen luokittajaan. Jos taas h_1 on liian suuri se johtaa liian sileään tiheysestimaattiin ja liian yksinkertaiseen luokittimeen.

5.4.5 Parzen-estimaatit luokituksessa

Parzen-estimaattien soveltaminen luokituksessa on suoraviivaista: Annettuna on kutakin luokkaa varten opetusnäytteet $\mathcal{D}_1, \ldots, \mathcal{D}_n$. Tällöin ensin estimoidaan (jos mahdollista) prioritodennäköisyydet $P(\omega_1), \ldots, P(\omega_c)$. Ja sitten estimoidaan luokkatiheydet $p(\cdot|\omega_i)$ Parzen-ikkunoiden avulla ja konstruoidaan Bayesin luokitin estimaattien perusteella. Vaikeutena on, että ei ole mahdollista laskea valmiiksi luokkatiheyksiä $p(\mathbf{x}|\omega_i)$ kaikille mahdollisille piirrevektoreille $\mathbf{x}$. Tästä syystä luokitettavalle

piirrevektorille $\mathbf{x}$ - jota kutsutaan *testivektoriksi* tai *testipisteeksi* - täytyy laskea ensin Parzen-estimaatti kunkin luokan luokkatiheydelle ja vasta sitten se voidaan luokittaa. Voimmekin kirjoittaa Parzen-luokittimen muodossa

$$\alpha_{Parzen}(\mathbf{x}) = \arg \max_{\omega_i, i=1,\ldots,c} \hat{P}(\omega_i) \frac{1}{n_i} \sum_{j=1}^{n_i} \frac{1}{h_i^d} \varphi(\frac{\mathbf{x} - \mathbf{x}_{i_j}}{h_i}), \tag{5.15}$$

jossa siis $\hat{P}(\omega_i)$ ovat prioritodennäköisyysestimaatit, $\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i}$ ovat luokkaan ω_i liittyvät opetusnäytteet, φ on ikkunafunktio (yleensä sama kaikille luokille) ja h_i on ikkunanleveys luokalle ω_i.

Kaavasta (5.15) voidaan havaita, että Parzen-luokittimet ovat laskennallisesti huomattavasti vaativampia kuin ML-estimointiin perustuvat luokittimet. Jokaisen testipisteen luokitus nimittäin vaatii nyt tiheysfunktioevaluoinnin jokaista harjoitusnäytettä kohti.

Huomaa, että sijoittamalla kaavaan (5.15) $\hat{P}(\omega_i) = n_i/n$, Parzen-luokittimeksi tulee

$$\alpha_{Parzen}(\mathbf{x}) = \arg \max_{\omega_i, i=1,\ldots,c} \frac{1}{n} \sum_{j=1}^{n_i} \frac{1}{h_i^d} \varphi(\frac{\mathbf{x} - \mathbf{x}_{i_j}}{h_i}).$$

5.4.6 Tilastolliset neuroverkot

Parzen luokittimet voidaan myös toteuttaa neuroverkko-konfiguraatiossa. Tällaista toteutusta kutsutaan *tilastolliseksi neuroverkoksi* (engl. Probabilistic neural network, PNN). Tällöin oletetaan, että opetusnäytteet on normalisoitu yksikönmittaisiksi, ts.

$$||\mathbf{x}_{i_j}|| = 1$$

ja myös testipiste täytyy näin ollen normalisoida. Tämä ei ole välttämättä järkevää läheskään kaikissa sovelluksissa. Merkitsemme näitä normalisoituja opetusnäytteitä symboleilla $\mathbf{y}_i$. Oletamme myös, että ikkunafunktio on normaalijakauman tiheysfunktio.

PNN koostuu d:stä *syöteyksiköstä*, n:stä *hahmoyksiköstä*, ja c:stä *luokkayksiköstä*. Kukin syöteyksikkö on kytketty kuhunkin hahmoyksikköön, ja kukin hahmoyksikkö on kytketty opetusnäytteen $\mathbf{x}_i$ luokan mukaiseen luokkayksikköön. Jokainen hahmoyksikkö laskee (normalisoidulle) syötteelle $\mathbf{x}$ sisätulon

$$net_i = \mathbf{y}_i^T \mathbf{x},$$

ja syöttää luokkayksikölle johon se on kytketty arvon $\exp[(net_i - 1)/h_n^2]$. Luokkayksiköt sitten summaavat niille syötetyt arvot ja valitsevat luokan, jolle tuo arvo on suurin. Tämä todellakin implementoi Parzen luokittimen, koska

$$\begin{aligned} \varphi(\frac{\mathbf{x} - \mathbf{y}_i}{h_n}) &\propto \exp[-(\mathbf{x} - \mathbf{y}_i)^T(\mathbf{x} - \mathbf{y}_i)/(2h_n^2)] && (5.16) \\ &= \exp[-(\mathbf{x}^T\mathbf{x} + \mathbf{w}_i^T\mathbf{w}_i - 2\mathbf{x}^T\mathbf{y}_i)/(2h_n^2)] && (5.17) \\ &= \exp[(net_i - 1)/h_n^2], && (5.18) \end{aligned}$$

jossa viimeinen yhtäsuuruus seuraa normalisointioletuksistamme. Merkintä $a(x) \propto b(x)$ tarkoittaa, että $a(x) = cb(x)$ jollain vakiolla c ja kaikilla x.

5.5 k:n lähimmän naapurin menetelmä

5.5.1 k_n lähimmän naapurin tiheysestimaatit

Parzen-ikkunoiden perusteella tapahtuvassa luokituksessa täytyi luokittimen suunnittelijan vetää hatusta sopivat ikkunafunktiot ja ikkunanpituudet. Yksi mahdollisuus on antaa näiden riippua harjoitusdatasta. Tiheysestimoinnin yleiseen formulointiin palataksemme, tämä tarkoittaa k_n:n kiinnittämistä ja tarvittavan lokeron tilavuuden V_n laskemista k_n:n perusteella. Tarkemmin, laskeaksemme estimaatin $p(\mathbf{x})$:lle keskitämme lokeron testipisteen $\mathbf{x}$ ympärille ja kasvatamme sitä niin kauan kunnes se sisältää k_n opetusnäytettä - $\mathbf{x}$:n k_n lähintä naapuria. Tässä k_n siis on ennalta annettu parametri. k_n lähimmän naapurin tiheysestimaatti n opetusnäytteen avulla on

$$p_n(\mathbf{x}) = \frac{k_n}{nV_n}, \tag{5.19}$$

missä V_n on pienimmän mahdollisen $\mathbf{x}$-keskisen pallon tilavuus, joka sisältää $\mathbf{x}$:n k_n lähintä naapuria. Jos siis todellinen tiheys on korkea $\mathbf{x}$:n lähistöllä, niin lokero $\mathcal{B}_n$ on pieni ja tiheysestimaatti on tarkka. Jos taas todellinen tiheys on matala, tiheysestimaatti ei ole niin tarkka. Englanniksi tätä estimaattia kutsutaan k_n *nearest neighbour* eli *KNN.* estimaatiksi.

Jotta KNN-tiheysestimaatti tarkentuisi pisteissä, joissa $p(\mathbf{x})$ on jatkuva vaaditaan, että

1. $k_n \to \infty$ kun $n \to \infty$;
2. $k_n/n \to 0$ kun $n \to \infty$.

Yllättävä KNN-estimaatin ominaisuus on, että vaikka se on jatkuva, niin sen osittaisderivaattojen ei tarvitse olla jatkuvia.

KNN-tekniikalla voidaan myös estimoida posterior todennäköisyyksiä $P(\omega_i|\mathbf{x})$. Tämä johtaa huomattavasti kiintoisampaan tulokseen kuin Parzen-ikkunoiden tapauksessa. Oletetaan taas siis, että opetusnäytteet $\mathcal{D}_1, \ldots, \mathcal{D}_c$ on kerätty sekoiteotannalla (kts. luku 4.1). Asetetaan nyt lokero, joka kattaa k näytettä (olivat ne mistä tahansa luokasta), testivektorin $\mathbf{x}$ ympärille. Näistä k_i kuuluu luokkaan ω_i. Tällöin estimaatti $\mathbf{x}$:n ja ω_i:n yhteistiheydelle on

$$p_n(\mathbf{x}, \omega_i) = \frac{k_i}{nV}.$$

Tästä normalisoimalla saadaan estimaatti posterior todennäköisyydelle

$$P_n(\omega_i|\mathbf{x}) = \frac{p_n(\mathbf{x}, \omega_i)}{p_n(\mathbf{x})} = \frac{p_n(\mathbf{x}, \omega_i)}{\sum_{j=1}^{c} p_n(\mathbf{x}, \omega_j)} = \frac{k_i}{\sum_{j=1}^{c} k_j} = \frac{k_i}{k}. \tag{5.20}$$

Posterior todennäköisyysestimaatti on maksimissaan kun testipisteen $\mathbf{x}$ luokaksi valitaan se, jolle k_i on suurin. Tämä on *k lähimmän naapurin luokitussääntö* (engl. k nearest neighbours rule), eli KNN-luokitussääntö.

5.5.2 Lähimmän naapurin sääntö

Tarkastellaan ensin *lähimmän naapurin luokitusta* , eli NN-luokitusta, joka on siis KNN-luokitussäännön erikoistapaus. Lähimmän naapurin luokituksessa - nimensä mukaisesti- valitaan $k = 1$ ja luokitetaan testipiste $\mathbf{x}$ sitä lähinnä olevan opetusnäytteen luokkaan. Ottamalla käyttöön kappaleessa 4.1 esitelty notaatio opetusnäytteille, luokitin voidaan kirjoittaa päätösfunktiomuodossa

$$\alpha_{nn}(\mathbf{x}) = \arg \min_{\omega_i, i=1,\ldots,c} \min_j ||\mathbf{x} - \mathbf{x}_{i_j}||. \tag{5.21}$$

Jotta KNN-tiheysestimaatit tarkentuisivat kohti todellista (luokka)tiheysfunktiota, olisi $k_n \to \infty$ kun opetusnäytteiden määrä kasvaa kohti ääretöntä. Tästä voidaan päätellä, että NN-luokitussääntö ei ole optimaalinen: Onhan $k_n = 1$ riippumatta opetusnäytteiden lukumäärästä. Voidaan kuitenkin osoittaa, että NN-luokittimen luokitusvirheen suuruus on korkeintaan kaksi kertaa Bayes virhe kun opetusnäytteiden määrä lähestyy ääretöntä. Tarkemmin

$$E(\alpha_{bayes}) \leq E(\alpha_{nn}) \leq E(\alpha_{bayes})(2 - \frac{c}{c-1}E(\alpha_{bayes})).$$

Luokitusvirheen yläraja on tässä tiukka, ja se saavutetaan kun todelliset luokkatiheysfunktiot ovat samat kaikille luokille.

Ylläoleva, varsin positiivinen, virhearvio on kuitenkin teoreettinen: Todellisuudessa meillä on vain äärellinen määrä opetusnäytteitä, eikä vastaavia (positiivisia) tuloksia äärelliselle opetusnäytemäärälle ole. Luokitin kuitenkin on yksinkertainen ja usein käytännössä hyvä. Luokittimen toimivuudelle on varsin yksinkertainen perustelu. Merkitään testipistettä $\mathbf{x}$ lähinnä olevaa opetusnäytettä $\mathbf{x}^*$:llä ja sen luokkaa ω^*:llä. Koska oletamme, että opetusnäytteet on luokitettu oikein, ω^* maksimoi todennäköisyyden $P(\omega_i|\mathbf{x}^*)$ luokkien $\omega_i, i = 1, \ldots, c$ suhteen suurella todennäköisyydellä. Tällöin, jos $\mathbf{x}$ on lähellä opetusnäytettä $\mathbf{x}^*$, niin ω^* maksimoi myös todennäköisyyden $P(\omega_i|\mathbf{x})$ luokkien $\omega_i, i = 1, \ldots, c$ suhteen suurella todennäköisyydellä.

Tarkastelemme vielä NN-luokittimen päätösalueiden geometriaa. Koska piirrevektorit luokitetaan samaan luokkaan kuin niitä lähinnä olevat opetusnäytteet, luokan ω_i päätösalue muodostuu soluista joiden pisteet ovat lähinnä tiettyä $\mathcal{D}_i$:n opetusnäytettä. Tällaista piirreavaruuden jakoa kutsutaan *Voronoi jaoksi*. Katso kuva 5.3, joka on Dudan, Hartin ja Storkin kirjasta.

5.5.3 k_n lähimmän naapurin sääntö

KNN säännössä siis etsitään testipisteen $\mathbf{x}$ k lähintä opetusnäytettä ja lasketaan mihin luokkaan näistä opetusnäytteestä kuuluu eniten näytteitä, johon sitten $\mathbf{x}$ luokitetaan. Tässä yhteydessä on hyvä korostaa, että KNN-sääntö - niinkuin kaikki muutkin päätössäännöt - on funktio piirreavaruudelta luokkien joukkoon, vaikkei KNN-sääntöä perinteisessä funktiomuodossa olekaan enää niin helppo esittää.

Algoritmi testipisteen $\mathbf{x}$ luokittamiseksi harjoitusnäytteiden

$$\mathcal{D}_i = \{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i}\}, i = 1, \ldots, c$$

on siis seuraava:

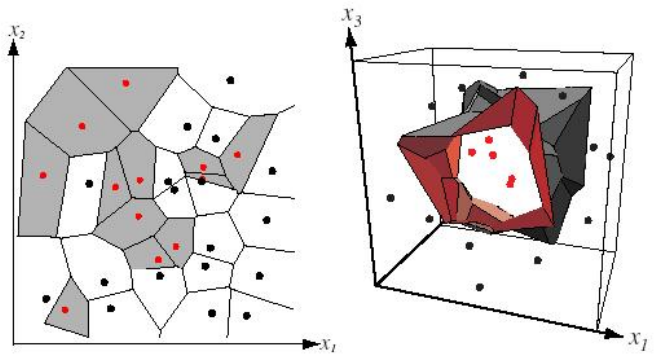


FIGURE 4.13. In two dimensions, the nearest-neighbor algorithm leads to a partitioning of the input space into Voronoi cells, each labeled by the category of the training point it contains. In three dimensions, the cells are three-dimensional, and the decision boundary resembles the surface of a crystal. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Kuva 5.3: Lähimmän naapurin luokittimen päätösalueet

1. Laske $\mathbf{x}$:n etäisyys jokaiseen opetusnäytteeseen; Merkitään näitä lukuja symboleilla d_{ij}, missä $i = 1, \ldots, c$ indikoi opetusnäytteen luokkaa ja $j = 1, \ldots, n_i$.

2. Etsi k pienintä etäisyyttä, merkitään näiden luokkia $a_1, a_2, \ldots, a_k$. Laske kuinka monta k:sta lähimmästä näytteestä kuuluu kuhunkin luokkaan, merkitään näitä k_i kuten edellä.

3. Luokita $\mathbf{x}$ suurimman k_i:n osoittamaan luokkaan.

Oletetaan esimerkiksi on meillä on yksiulotteinen opetusdata

ω_1	2	20	5	11	15
ω_2	21	31	34	21	10
ω_3	28	26	26	24	30

ja haluamme luokittaa pisteen $\mathbf{x} = 20$. Etäisyydet d_{ij} ovat

ω_1	18	0	15	9	5
ω_2	1	11	14	1	10
ω_3	8	6	6	4	10

ja siis $a_1 = \omega_1, a_2 = \omega_2, a_3 = \omega_2$. Tällöin, $k_1 = 1, k_2 = 2, k_3 = 0$ ja näyte luokitetaan luokkaan ω_2.

Kahden luokan tapauksessa KNN-säännölle on voimassa seuraava virhearvio kun opetusnäytteitä on mielivaltaisen monta:

$$E(\alpha_{bayes}) \leq E(\alpha_{knn}) \leq E(\alpha_{bayes}) + \sqrt{\frac{2E(\alpha_{nn})}{k}}.$$

Tämänkin virhearvion nojalla voidaan todeta, että kun $k \to \infty$, niin KNN-luokitin on optimaalinen. Kuitenkin taas käytäntö on erilainen: varsinkin silloin kun piirreavaruus on korkeadimensioinen, tarvittava opetusnäytteiden määrä kasvaa hyvin nopeasti k:n mukana.

KNN-luokitin, samoin kuin NN-luokitin, kärsii samasta käytännön vaikeudesta, josta mainittiin jo Parzen luokittimen yhteydessä. Nimittäin luokituksen laskennallinen vaativuus kasvaa opetusnäytteiden määrän mukana, koska luokittimen täytyy laskea testipisteen etäisyys jokaiseen opetusnäytteeseen. Tämä saattaa muodostua ongelmaksi, jos luokittimen täytyy olla hyvin nopea. Mainittakoon kuitenkin, että monia menetelmiä KNN-luokitinten nopeuttamiseksi on kehitetty.

5.5.4 Metriikoita

Olemme tähän asti olettaneet, että piirreavaruuden pisteiden (vektorien) $\mathbf{a}$ ja $\mathbf{b}$ läheisyyttä mitataan Euklidisen metriikan avulla

$$d(\mathbf{a}, \mathbf{b}) = ||\mathbf{a} - \mathbf{b}|| = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2}.$$

Näin ei kuitenkaan tarvitse olla. Monia muitakin etäisyysmittoja, eli metriikoita, voidaan käyttää. Formaalisti metriikka määritellään kaksipaikkaisena kuvauksena $d : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Tällä kuvauksella oletetaan olevan seuraavat ominaisuudet:

1. $d(\mathbf{a}, \mathbf{b}) \geq 0$,
2. $d(\mathbf{a}, \mathbf{b}) = 0$ jos ja vain jos $\mathbf{a} = \mathbf{b}$,
3. $d(\mathbf{a}, \mathbf{b}) = d(\mathbf{b}, \mathbf{a})$,
4. $d(\mathbf{a}, \mathbf{b}) + d(\mathbf{b}, \mathbf{c}) \geq d(\mathbf{a}, \mathbf{c})$,

missä $\mathbf{a}, \mathbf{b}, \mathbf{c}$ ovat mielivaltaisia avaruuden $\mathbb{R}^d$ vektoreita. Hahmontunnistuksessa on joskus tarpeellista tinkiä ominaisuudesta 2, muutoin metriikkojen ominaisuudet ovat tärkeitä hahmontunnistuksenkin kannalta.

Esimerkkejä hyödyllistä metriikoista ovat

- Minkowskin metriikat:

$$L_m(\mathbf{a}, \mathbf{b}) = (\sum_{i=1}^{d} (|a_i - b_i|^m)^{1/m}.$$

- L-ääretön metriikka

$$L_\infty(\mathbf{a}, \mathbf{b}) = \max_i |a_i - b_i|.$$

- Mahalanobis-etäisyys

$$d_{Mahalanobis,C}(\mathbf{a}, \mathbf{b}) = \sqrt{(\mathbf{a} - \mathbf{b})^T C^{-1} (\mathbf{a} - \mathbf{b})},$$

  missä C on annettu positiividefiniitti matriisi.

5.6 Virhelähteistä

Tässä luvussa on monta kertaa todettu, että teoreettiset luokittimien optimaalisuustulokset pätevät vain jos opetusnäytteitä on ääretön määrä. Kuitenkin käytännössä opetusnäytteitä on äärellinen määrä ja tässä kappaleessa erittelemmekin käytännön luokitinten virhelähteitä.

Ennen virhelähteisiin puuttumista, huomautamme että hahmontunnistuksessa kannattaa erottaa kaksi eri tyypin virhettä. *Opetusvirhe* tarkoittaa luokitusvirhettä opetusnäytteille, ts. kuinka suuri osa opetusnäytteistä luokittuu väärin suunnitellussa luokittimessa. Tämän tyypin virheestä emme nyt ole kiinnostuneita. Huomattavasti tärkeämpi on *testivirhe*, joka kuvaa kuinka suuri osa kaikista mahdollisista piirrevektoreista luokitetaan väärin. Kannattaa huomata, että Bayes-luokitin minimoi nimenomaan testivirheen kun vaaditut oletukset ovat voimassa. Todellisen testivirheen arviointi on luonnollisesti vaikeaa ja siihen palaamme vielä kurssin aikana.

Vielä ennen kuin repostelemme virhelähteitä - siis testivirheen lähteitä - kertaamme mitä erilaiset luokittimet halusivat syötteenään. Ensinnäkin, kaikki ohjatut luokittimet - joita siis tässä luvussa esitetyt luokittimet ovat - vaativat syötteekseen opetusnäytteitä. ML-estimaatteihin perustuvat luokittimet tarvitsevat lisäksi tiedon luokkatiheysfunktioiden parametrisista muodoista. Parzen-luokittimet vaativat tiedon ikkunafunktioista ja ikkunanleveyksistä. KNN-luokittimia varten taas täytyy valita sopiva k_n.

Luonnollisestikin kuhunkin luokitusongelmaan liittyy välttämätön virhelähteensä, joka on ongelmalle ominainen, eikä riipu käytettävästä luokittimesta. Tämä virhe on yhtä kuin ongelman *Bayes-virhe*. Tätä virhettä ei voi millään tapaa vähentää luokitusongelmaa muuttamatta. Jos otetaan koko hahmontunnistusjärjestelmä huomioon, niin luokitusongelmaa on mahdollista muuttaa esimerkiksi kasvattamalla käytettävien piirteiden määrää.

Toinen syy luokitusvirheisiin on, että ongelmalle valittu malli on väärä. Kutsumme tätä *mallivirheeksi*. ML-luokituksessa mallivirheitä tapahtuu, jos luokkatiheysfunktioille valitut parametriset muodot eivät olekaan oikeita, ts. luokasta ω_i tuleva data ei olekaan jakautunut minkään tiheyksistä $p(\cdot|\omega_i, \theta_i)$ mukaisesti. Mallivirheiksi epäparametrisissa luokittimissa voidaan tulkita ei optimaalinen ikkunafunktioiden, ikkunanleveyksien tai parametrin k_n valinta. Mallivirheet ovat suuremmassa roolissa ML-luokituksessa ja epäparametrisissa luokittimissa muut virhetyypit ovat useimmiten tärkeämpiä. Mallivirheitä voidaan vähentää valitsemalla datalle entistä parempi (tilastollinen) malli. (Tämä ei kuitenkaan yleensä ole aivan yksinkertaista.)

Kolmas ja viimeinen virhelähde on *estimointivirhe*. Tämä virhe seuraa siitä, että luokitinten opettamiseen on käytössä vain äärellinen määrä opetusnäytteitä. Epäparametrisessa tapauksessa opetusnäytteiden määrällä on suuri merkitys. Jos toisaalta ML-luokittimissa luokkatiheyksien malli on yksinkertainen, niin opetusnäytteitä ei tarvita kohtuuttoman montaa. Mitä monimutkaisempi malli sitä enemmän opetusnäytteitä tarvitaan, eli sopivan tyyppisen luokittimen valinta on monesti tasapainoilua pienen estimointivirheen ja pienen mallivirheen välillä. Estimointivirhettä voidaan pienentää lisäämällä opetusaineiston kokoa.

Yleisesti voidaan vielä sanoa, että mitä enemmän piirteitä on käytössä, sitä suu-

rempi estimointivirhe on. Lisäksi opetusnäytteiden tarve kasvaa - erityisesti epäparametrisessa tapauksessa - eksponentiaalisesti piirteiden määrään verrattuna. Tätä ilmiötä kutsutaan usein *dimensiokiroukseksi* (engl. curse of dimensionality).

Luku 6

Lineaariset erotinfunktiot ja luokittimet

6.1 Johdanto

Tähän asti olemme suunnitelleet luokittimia estimoimalla luokkatiheysfunktioita ja prioritodennäköisyyksiä harjoitusdatan perusteella. Estimoitujen luokkatiheysfunktioiden ja priorien perusteella olemme sitten johtaneet Bayes luokittimen erotinfunktiot. Näin saatu luokitin minimoi luokitusvirheen kunhan vain luokkatiheysestimaatit ovat lähellä todellisia luokkatiheyksiä ja estimoidut priorit ovat myös lähellä todellisia prioreita.

Ajatellaan nyt toista tapaa johtaa luokitin: Oletetaan, että tunnemme erotinfunktioiden parametriset muodot, samaan tapaan kuin ML-estimointi kappaleessa. Tällöin yritämme suoraan löytää erotinfunktiot estimoimatta välissä luokkatiheyksiä. Tämän tavan hyvät puolet voi sisäistää yksinkertaisen esimerkin avulla: Oletetaan, että meillä on kahden luokan luokitusongelma, jossa luokkatiheysfunktiot ovat normaalijakautuneita ja niiden kovarianssimatriisit ovat yhtä suuria. Tällöin, kuten kappaleessa 4.6 todettiin, meillä on lineaarinen luokitin joka voidaan esittää (yhden) erotinfunktion avulla:

$$g(\mathbf{x}) = g_1(\mathbf{x}) - g_2(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0,$$

missä

$$\mathbf{w} = \Sigma^{-1}(\mu_1 - \mu_2)$$

ja

$$w_0 = -\frac{1}{2}(\mu_1 - \mu_2)^T\Sigma^{-1}(\mu_1 - \mu_2) + \ln P(\omega_1) - \ln P(\omega_2).$$

Tämän erotinfunktion johtamiseen oli tarpeellista estimoida $2 \cdot d + (d^2 + d)/2 + 1$ parametrien arvoa. Ensimmäinen termi tässä juontaa kahdesta keskiarvovektorista, toinen yhdestä (symmetrisestä) kovarianssimatriisista ja kolmas kahdesta prioritodennäköisyydestä, joiden summa on yksi. Mutta voisimmeko estimoida *painovektoria* $\mathbf{w}$ ja *kynnyspainoa* w_0 suoraan? Tässä tapauksessa täytyisi estimoida ainoastaan

$d+1$ parametrien arvoa. Tämän lisäksi, erotinfunktioihin kohdistuva parametrisointi voi joskus olla luontevampi kuin tiheysfunktioiden parametrisointi.

Vastaus on positiivinen: voimme estimoida suoraan $\mathbf{w}$:tä ja w_0:llaa minimoimalla harjoitusvirhettä tai jotakin harjoitusvirheen funktiota. Heti kuitenkin huomautamme, että tällöin menetämme suoran kontaktin testivirheeseen, joka - kuten todettua - on harjoitusvirhettä huomattavasti tärkeämpi suure.

Tässä luvussa tarkastelemme proseduureja, joiden avulla voimme estimoida lineaarisia erotinfunktioita suoraan ja tutustumme myös minkälaisiin päätösalueisiin nämä johtavat.

6.2 Lineaaristen luokittimien ominaisuuksia

6.2.1 Lineaariset luokittimet

Palautetaan aluksi mieleen, miten luokitin määriteltiin erotinfunktioiden avulla. Jokaiselle luokalle oli määritelty erotinfunktio $g_i, i = 1, \ldots, c$, joka otti syötteekseen piirrevektorin $\mathbf{x}$. Nyt $\mathbf{x}$ luokitetaan luokkaan ω_i, jos

$$g_i(\mathbf{x}) > g_j(\mathbf{x})$$

kaikille $i \neq j$. (Jos $g_i(\mathbf{x}) = g_j(\mathbf{x})$ voimme turvautua johonkin sattumanvaraiseen sääntöön kuten luokittamaan $\mathbf{x}$ luokkaan ω_i, jos $i < j$. Tällä ei ole suurta merkitystä kuten ei ollut Bayes luokittimenkaan tapauksessa.)

Erotinfunktio g_i on lineaarinen, jos

$$g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_{i0} = \sum_{j=1}^{d} w_{ij}x_j + w_{i0},$$

missä $\mathbf{w}_i = [w_{i1}, \ldots, w_{id}]^T$ on painovektori ja skalaari w_{i0} on kynnyspaino. Luokitinta, joka voidaan esittää lineaaristen erotinfunktioiden avulla kutsutaan lineaariseksi luokittimeksi.

6.2.2 Kaksi luokkaa

Tarkastellaan nyt lineaarista luokitinta kahden luokan luokitusongelman ratkaisemiseksi. Pisteet $\mathbf{x}$ sen päätöspinnalla toteuttavat yhtälön

$$g_1(\mathbf{x}) = \mathbf{w}_1^T\mathbf{x} + w_{10} = \mathbf{w}_2^T\mathbf{x} + w_{20} = g_2(\mathbf{x}),$$

eli

$$(\mathbf{w}_1 - \mathbf{w}_2)^T\mathbf{x} + w_{10} - w_{20} = 0.$$

Merkitään nyt $\mathbf{w} = \mathbf{w}_1 - \mathbf{w}_2$ ja $w_0 = w_{10} - w_{20}$, eli esitetään luokitin yhden erotinfunktion $g(\mathbf{x})$ avulla. Saamme päätöspinnaksi:

$$\mathcal{R}_{12} = \{\mathbf{x} | \mathbf{w}^T\mathbf{x} + w_0 = 0\}. \tag{6.1}$$

Tämä päätöspinta on hypertaso: Kun $d = 1$ se on piste, kun $d = 2$ se on suora, ja niin edelleen.

Valitaan kaksi pistettä päätöspinnalta, sanotaan $\mathbf{x}_1$ ja $\mathbf{x}_2$. Nyt

$$\mathbf{w}^T\mathbf{x}_1 - \mathbf{w}^T\mathbf{x}_2 - w_0 + w_0 = \mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = 0.$$

Joten painovektori $\mathbf{w}$ on päätöspinnan normaali. Edelleen voimme kirjoittaa

$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{||\mathbf{w}||},$$

jossa $\mathbf{x}_p$ on piirrevektorin $\mathbf{x}$ (normaali)projektio päätöspinnalle $\mathcal{R}_{12}$. Tällöin, koska $g(\mathbf{x}_p) = 0$, saamme

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = \mathbf{w}^T(\mathbf{x}_p + r\frac{\mathbf{w}}{||\mathbf{w}||}) + w_0 = r||\mathbf{w}||.$$

Skalaari r on siis positiivinen jos $\mathbf{x}$ kuuluu luokkaan ω_1 ja negatiivinen jos $\mathbf{x}$ kuuluu luokkaan ω_2. Huomaamme myös, että itseisarvoltaan suuret erotinfunktion arvot tarkoittavat, että piirrevektori $\mathbf{x}$ on kaukana päätöspinnasta. Kuva 6.1 selventää tilannetta. Se on kuva 5.2 Dudan, Hartin ja Storkin kirjasta.

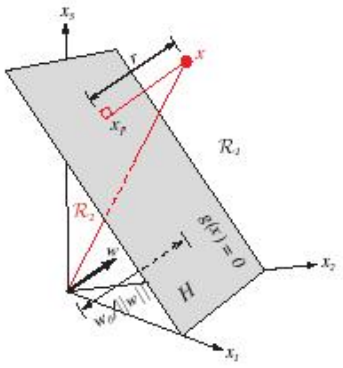


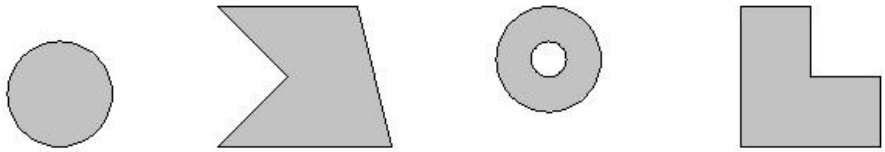
FIGURE 5.2. The linear decision boundary H, where $g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 = 0$, separates the feature space into two half-spaces $\mathcal{R}_1$ (where $g(\mathbf{x}) > 0$) and $\mathcal{R}_2$ (where $g(\mathbf{x}) < 0$). From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Kuva 6.1: Lineaarisen luokittimen geometria

6.2.3 Useampi luokka

Edellisen kappaleen päätöspintoja koskevat tulokset tietenkin pätevät myös useamman luokan tapauksessa, jos tarkastelemme vain luokkien ω_i ja ω_j välistä päätöspintaa. Otettaessa kaikki luokat huomioon, huomaamme, että päätöspinnat ovat nyt hyperpintojen osia. Päätösalueistakin voidaan sanoa paljon: Ne ovat nimittäin *konvekseja*.

Joukko $R \subset \mathbb{R}^d$ on konveksi (kts. kuva 6.2), jos kaikilla $\mathbf{r}_1, \mathbf{r}_2 \in R$ ja $(1-\lambda)\mathbf{r}_1 + \lambda\mathbf{r}_2 \in R$, kun $\lambda \in [0,1]$. Eli selväkielellä: joukko R on konveksi ,jos jokainen sen pisteiden $\mathbf{r}_1$ ja $\mathbf{r}_2$ välinen jana kuuluu myös kokonaisuudessaan joukkoon R. Esimerkiksi pallo on kolmiulotteisessa avaruudessa konveksi joukko. Konveksit joukot ovat yhdesti yhtenäisiä, joten siis lineaaristen luokitinten päätösalueet ovat yhdesti yhtenäisiä.



Kuva 6.2: Ylläolevista alueista ainoastaan vasemmanpuolimmainen on konveksi, kolme muuta eivät ole konvekseja.

Lineaaristen luokitinten päätösalueiden konveksisuus voidaan todistaa kun ajatellaan kahta päätösalueeseen $\mathcal{R}_i$ kuuluvaa pistettä $\mathbf{r}_1, \mathbf{r}_2$. Näille pisteille siis

$$g_i(\mathbf{r}_1) > g_j(\mathbf{r}_1), g_i(\mathbf{r}_2) > g_j(\mathbf{r}_2) \tag{6.2}$$

kaikille $j \neq i$. Nyt, jos tarkastellaan pistettä $(1-\lambda)\mathbf{r}_1+\lambda\mathbf{r}_2$, voidaan erotinfunktioiden lineaarisuuden nojalla päätellä, että

$$g_i((1-\lambda)\mathbf{r}_1 + \lambda\mathbf{r}_2) = \mathbf{w}_i^T((1-\lambda)\mathbf{r}_1 + \lambda\mathbf{r}_2) + w_{i0} > g_j((1-\lambda)\mathbf{r}_1 + \lambda\mathbf{r}_2). \tag{6.3}$$

Siis päätösalueet ovat konvekseja! Kaavan (6.3) totuudenmukaisuuden osoittaminen ehtojen (6.2) ollessa voimassa säästetään harjoitustehtäväksi.

Päätösalueiden konveksisuus luonnollisesti rajoittaa niiden luokitusongelmien määrää, jotka voidaan ratkaista lineaaristen luokitinten avulla.

6.3 Lineaarisesti separoituvat harjoitusnäytteet

Tässä kappaleessa tarkastelemme minkälaisia ongelmia lineaaristen luokitinten voidaan olettaa ratkaisevan. Tarkastelemme ensin kahden luokan tapausta. Oletamme, että luokitusongelmamme on sellainen että on olemassa lineaarinen luokitin, jonka luokitusvirhe on hyvin pieni. Meillä on taaskin näytteitä luokista ω_1 ja ω_2. Näitä merkitään edelleen

$$\mathcal{D}_i = \{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i}\}, i = 1, 2,$$

missä n_i on luokkaan ω_i kuuluvien näytteiden määrä. Jos nyt on olemassa sellainen lineaarinen erotinfunktio, jolle

$$g(\mathbf{x}_{1j}) > 0,$$

kaikille $j = 1, \ldots, n_1$ ja

$$g(\mathbf{x}_{2j}) < 0$$

kaikille $j = 1, \ldots, n_2$, sanomme, että näyttejoukot $\mathcal{D}_1$ ja $\mathcal{D}_2$ ovat *lineaarisesti separoituvia.*

Ylläoleva ehto ja kahden luokan lineaariset luokittimet voidaan esittää myös kätevämmässä muodossa. Tätä varten teemme kaksi modifikaatiota notaatioon:

1) Jatkettu piirrevektori (engl. augmented feature vector) $\mathbf{y}$ muodostetaan piirrevektorista $\mathbf{x}$ seuraavasti:

$$\mathbf{y} = [1, x_1, \ldots, x_d]^T.$$

Ja samoin jatkettu painovektori

$$\mathbf{a} = [w_0, w_1, \ldots, w_d]^T = [w_0, \mathbf{w}^T]^T.$$

Lineaarinen erotinfunktio voidaan nyt esittää muodossa

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = \mathbf{a}^T\mathbf{y} = g(\mathbf{y}).$$

2) Merkitsemme opetusnäytteestä $\mathbf{x}_{ij}$ generoitua jatkettua opetusnäytettä symbolilla $\mathbf{y}_{ij}$. Kahden luokan tapauksessa pääsemme eroon kahdesta näyteluokasta $\mathcal{D}_1, \mathcal{D}_2$ kertomalla luokan ω_2 jatketut opetusnäytteet luvulla -1. Tämä koska

$$\mathbf{a}^T\mathbf{y}_{2j} < 0 \Leftrightarrow \mathbf{a}^T(-\mathbf{y}_{2j}) > 0.$$

Täten voimme nyt 'unohtaa' luokan ω_2 ja kertoa siihen kuuluvat harjoitusnäytteet -1:llä ja liittää ne luokan ω_1 harjoitusnäytteiden joukkoon. Näin saatua harjoitusnäytejoukkoa merkitsemme ytimekkäästi $\mathcal{D} = \{\mathbf{y}_1, \ldots, \mathbf{y}_{n_1+n_2}\}$. Huomaa, että kertominen -1:llä täytyy tehdä nimenomaan jatketuille opetusnäytteille. (Miksi?)

Esimerkki. Meillä on opetusnäytteet

$$\mathcal{D}_1 = \{[1,1]^T, [2,2]^T, [2,1]^T\}$$

ja

$$\mathcal{D}_2 = \{[1,-1]^T, [1,-2]^T, [2,-2]^T]$$

Tällöin modifioitu joukko jatkettuja näytteitä on

$$\mathcal{D} = \{[1,1,1]^T, [1,2,2]^T, [1,2,1]^T, [-1,-1,1]^T, [-1,-1,2]^T, [-1,-2,2]^T\}.$$

Kun nämä kaksi muunnosta on tehty, kahden luokan lineaarinen separoituvuus voidaan määritellä: Harjoitusnäytejoukko $\mathcal{D}$ on lineaarisesti separoituva, jos on olemassa sellainen jatkettu painovektori $\mathbf{a}$, että

$$\mathbf{a}^T\mathbf{y}_j > 0 \tag{6.4}$$

kaikille $\mathbf{y}_1, \ldots, \mathbf{y}_{n_1+n_2}$.

Jos nyt oletamme, että harjoitusnäytteet ovat lineaarisesti separoituvia, niin luokittimen suunnittelemiseksi voimme yrittää ratkaista epäyhtälöryhmän (6.4), yksi mahdollinen ratkaisumenetelmä esitetään seuraavassa kappaleessa. Huomautetaan kuitenkin sitä ennen, että ratkaisut eivät ole yksikäsitteisiä. Ensinnäkin, jos meillä on vektori $\mathbf{a}$ joka toteuttaa epäyhtälöryhmän (6.4), niin voimme aina kertoa sen positiivisella vakiolla (muista, että painovektorit olivat päätöspintojen normaaleja). Toiseksi, mahdollisia päätöspintoja voi olla useita, jolloin meillä siis on useita 'aidosti' erilaisia jatkettuja painovektoreita, jotka ratkaisevat luokitusongelman harjoitusnäytteille.

6.4 Perceptron kriteeri ja algoritmi kahden luokan tapauksessa

6.4.1 Perceptron kriteeri

Löytääksemme painovektorin $\mathbf{a}$, joka toteuttaa epäyhtälöt

$$\mathbf{a}^T\mathbf{y}_j > 0,$$

kaikilla j, formuloimme ongelman optimointiongelmaksi. Etsimme funktion

$$J_p(\mathbf{a}) = \sum_{j:\mathbf{a}^T\mathbf{y}_j \leq 0} -\mathbf{a}^T\mathbf{y}_j \tag{6.5}$$

minimiä. Tätä funktiota kutsumme *Perceptron kriteeriksi*. Siis: summaamme yli painovektorin $\mathbf{a}$ määrittämän luokittimen väärin luokittamien opetusnäytteiden ja laskemme yhteen näiden opetusnäytteiden pistetulot vektorin $-\mathbf{a}$ kanssa. Huomaa, että on täysin sopimuksenvaraista luokittuuko jatkettu piirrevektori $\mathbf{y}_j$ jolle $\mathbf{a}^T\mathbf{y}_j = 0$ oikein vai väärin. Tässä oletamme, että se luokittuu väärin. Tämä oletus on myös hyödyllinen ratkaisun $\mathbf{a} = \mathbf{0}$ välttämikseksi [1].

Perceptron kriteerin arvo $J_p(\mathbf{a})$ on aina positiivinen. Se saa miniminsä, joka siis on nolla, kun painovektori $\mathbf{a}$ toteuttaa epäyhtälöryhmät (6.4). (Ja myös degeneraatissa tapauksessa $\mathbf{a} = \mathbf{0}$, joka ei ole haluamamme tulos). Näin siis on epäyhtälöiden ratkaisu muunnettu minimointiongelmaksi. Muistetaan kuitenkin vielä kerran, että epäyhtälöryhmällä (6.4) voi olla useita ratkaisuja $\mathbf{a}$ ja kaikille näille $J_p(\mathbf{a}) = 0$.

Funktio $\sum_{j:\mathbf{a}^T\mathbf{y}_j<0} -1$ tietenkin myös täyttäisi edelläesitetyt vaatimukset. Tämä funktio ei kuitenkaan ole jatkuva, joten se on vaikea minimoida numeerisesti. J_p taas on jatkuva.

[1] Tällä sopimuksella ei ole merkitystä Perceptron kriteerin arvon kannalta, mutta sillä on merkitystä määritettäessä Percepteron kriteerin gradienttia. Gradientin määritystä tarvitaan Perceptron algoritmin määrittämiseen.

6.4.2 Perceptron algoritmi

Yleensä funktion J_p analyyttinen minimointi ei ole mahdollista ja meidän täytyy turvautua numeerisiin ratkaisumenetelmiin. Yksinkertaisin numeerinen tapa ratkaista minimointiongelmia on *gradienttimenetelmä* (engl. steepest descent method). Tässä valitaan jokin alkuarvo painovektorille $\mathbf{a}$ (merkitsemme tätä $\mathbf{a}(0)$) ja lähdetään päivittämään $\mathbf{a}$:ta funktion J_p gradienttia vastakkaiseen suuntaan pisteessä $\mathbf{a}(0)$. Tämä siksi, että minkä tahansa jatkuvan ja derivoituvan funktion f arvo vähenee voimakkaimmin suuntaan $-\nabla f(\mathbf{a})$ pisteessä $\mathbf{a}$. Perceptron kriteerin J_p gradientti pisteessä $\mathbf{a}$ on

$$\nabla J_p(\mathbf{a}) = \sum_{j:\mathbf{a}^T\mathbf{y}_j \leq 0} -\mathbf{y}_j. \tag{6.6}$$

Joten päivitys-säännöksi muodostuu:

$$\mathbf{a}(t+1) = \mathbf{a}(t) + \eta \sum_{j:\mathbf{a}(t)^T\mathbf{y}_j \leq 0} \mathbf{y}_j,$$

missä t on iteraatiokierroksen numero ja η on askelpituus. Tästä saamme Perceptron algoritmin:

Perceptron algoritmi
Aseta $t \leftarrow 0$, alusta $\mathbf{a}(0), \eta, \epsilon$.
while $-\sum_{j:\mathbf{a}(t)^T\mathbf{y}_j \leq 0} \mathbf{a}^T\mathbf{y}_j > \epsilon$ **do**
  $\mathbf{a}(t+1) = \mathbf{a}(t) + \eta \sum_{j:\mathbf{a}(t)^T\mathbf{y}_j \leq 0} \mathbf{y}_j$.
  Aseta $t \leftarrow t+1$
end while
Palauta $\mathbf{a}(t)$.

Algoritmille täytyy alustuksen lisäksi antaa kaksi parametria: pysähtymisehto ϵ ja askelpituus η. Kunhan vain harjoitusnäytteet ovat lineaarisesti separoituvia, ϵ voidaan valita nollaksi. Algoritmi nimittäin pysähtyy äärellisessä määrässä askelia kun harjoitusnäytteet ovat lineaarisesti separoituvia. Jos harjoitusnäytteet eivät ole lineaarisesti separoituvia, Perceptron algoritmi ei konvergoi ja siksi se ei ole kovin hyvä valinta tässä tapauksessa. Askelpituus η, jota hahmontunnistuksessa usein kutsutaan oppimisnopeudeksi, voidaan valita esimerkiksi ykköseksi. Mainittakoon vielä, että algoritmin lopputulos riippuu alustuksesta $\mathbf{a}(0)$.

Esimerkki. Nyt katsomme miten edellisen kappaleen esimerkin harjoitusjoukolle generoidaan lineaarinen luokitin Perceptron algoritmin avulla. Valitaan $\mathbf{a}(0) = [1,1,1]^T$. Ensimmäisellä kierroksella väärin luokittuvat näytteet 4,5 ja 6. Joten,

$$\mathbf{a}(1) = \mathbf{a}(0) + \mathbf{y}_4 + \mathbf{y}_5 + \mathbf{y}_6 = [-2,-3,6]^T.$$

Sitten väärin luokittuu näyte 3, siitä

$$\mathbf{a}(2) = \mathbf{a}(1) + \mathbf{y}_3 = [-1,-1,7]^T,$$

joka on myös lopputulos. Luokitussäännöksi muodostuu: Luokita $\mathbf{x}$ luokkaan ω_1 jos $g(\mathbf{x}) = [-1, 7]\mathbf{x} - 1 > 0$ ja luokkaan ω_2 muulloin. Jatketun painovektorin perusteella on myös helppo määrittää luokittimen päätöspinta. Koska luokitin on lineaarinen ja $d = 2$, päätöspinta on suora jä päätospinnalla oleville piirrevektoreille $\mathbf{x}$ pätee $g(\mathbf{x}) = 0$. Helpoin tapa suoran piirtämiseksi koordinaatistoon on ratkaista kaksi eri pistettä suoralta: tämähän määrittää suoran täysin. Esimerkiksi piirrevektorit $[-1, 0]^T$ ja $[6, 1]^T$ ovat päätöspinnalla, joten päätöspintaa kuvaava suora voidaan piirtää niiden välille (ks. Kappaleen 6.6 kuva 6.3).

6.5 Perceptron useammalle luokalle

Tähän asti olemme keskittyneet ainoastaan kahden luokan lineaaristen luokitinten suunnitteluun. Nyt oletamme, että haluamme suunnitella c:n luokan lineaarisen luokittimen. Tätä varten meillä on (jatkettuja) harjoitusnäytteitä c:sta luokasta:

$$\mathcal{D}_i = \{\mathbf{y}_{i1}, \ldots, \mathbf{y}_{in_i}\}, i = 1, \ldots, c.$$

Sanomme, että nämä näytejoukot ovat lineaarisesti separoituvia jos on olemassa sellainen lineaarinen luokitin joka luokittaa kaikki näytteet korrektisti. Tehtävänä nyt on siis etsiä sellaiset painovektorit $\mathbf{a}_1, \ldots, \mathbf{a}_c$, että

$$\mathbf{a}_i^T \mathbf{y} > \mathbf{a}_j^T \mathbf{y} \tag{6.7}$$

kaikille $j \neq i$ aina kun $\mathbf{y} \in \mathcal{D}_i$. Toisin sanoen

$$\mathbf{a}_i^T \mathbf{y}_{ik} > \mathbf{a}_j^T \mathbf{y}_{ik},$$

kaikille $i = 1, \ldots, c, j \neq i$ ja $k = 1, \ldots, n_i$. Nämä yhtälöt voidaan niin kutsutun *Keslerin konstruktion* avulla manipuloida kahden luokan ja $c \cdot d$ piirteen ongelmaksi. Tämä on hyödyllinen teoreettinen työkalu, mutta laskennallisesti se ei ole tehokasta.

Annamme nyt Perceptron algoritmin useammalle luokalle:

Perceptron algoritmi c:lle luokalle
Aseta $t \leftarrow 0$, alusta $\mathbf{a}_1(0), \ldots, \mathbf{a}_c(0)$.
while Väärin luokittettuja opetusnäytteitä olemassa **do**
  for all väärin luokitetuille $\mathbf{y}_{ik}$ **do**
    $\mathbf{a}_i(t+1) = \mathbf{a}_i(t) + \mathbf{y}_{ik}$
    $\mathbf{a}_j(t+1) = \mathbf{a}_j(t) - \mathbf{y}_{ik}$ jos $\mathbf{a}_i(t)^T \mathbf{y}_{ik} \leq \mathbf{a}_j(t)^T \mathbf{y}_{ik}$
    $\mathbf{a}_l(t+1) = \mathbf{a}_l(t)$ jos $\mathbf{a}_i(t)^T \mathbf{y}_{ik} > \mathbf{a}_l(t)^T \mathbf{y}_{ik}$
    Aseta $t \leftarrow t + 1$
  end for
end while
Palauta $\mathbf{a}_1(t), \ldots, \mathbf{a}_c(t)$.

Kuten kahdenkin luokan tapauksessa tämä algoritmi toimii vain jos opetusnäytteet ovat lineaarisesti separoituvia.

6.6 Pienimmän neliösumman kriteeri

Tarvitsemme vielä sopivan algoritmin lineaaristen luokitinten suunnitteluun, kun opetusnäytteet eivät ole lineaarisesti separoituvia. Tällainen voidaan perustaa pienimmän neliösumman kriteeriin. Käymme nyt kursorisesti läpi pienimmän neliösumman kriteerin kahden luokan tapauksessa.

Teemme taas kaksi kappaleessa 6.3 mainittua modifikaatiota opetusnäytteille, jonka tuloksena meillä on yksi joukko laajennettuja opetusnäytteitä. Pyrimme nyt minimoimaan funktion

$$J_s(\mathbf{a}) = \sum_{i=1}^{n_1+n_2} (\mathbf{a}^T \mathbf{y}_i - 1)^2. \tag{6.8}$$

Tämä funktio voidaan myös kirjoittaa matriisimuodossa

$$J_s(\mathbf{a}) = ||Y\mathbf{a} - \mathbf{1}||^2, \tag{6.9}$$

jossa matriisi Y on $n_1 + n_2 \times d + 1$ matriisi sisältäen tiedon opetusnäytteistä ja $\mathbf{1}$ pelkästään ykkösiä alkioina sisältävä $n_1 + n_2$ paikkainen pystyvektori. Tarkemmin

$$Y = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_{n_1+n_2} \end{bmatrix}$$

ja

$$\mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

Tällä ongelmalla on ratkaisu

$$\mathbf{a} = (Y^T Y)^{-1} Y^T \mathbf{1}. \tag{6.10}$$

Huomaa, ettei oletuksia Y:n ei-singulaarisuudesta tai neliöllisyydestä tarvita. Tässä esiintyneen $\mathbf{1}$ vektorin valinta perustuu relaatioon Bayes luokittimen kanssa. Tästä lisää Dudan, Hartin ja Storkin kirjan kappaleessa 5.8.3.

Esimerkki. Demonstroidaan vielä kappaleen 6.3 esimerkin harjoitusnäytteiden avulla luokittimen suunnittelua perustuen pienimmän neliösumman kriteerin avulla. Tällöin

$$Y = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 1 \\ -1 & -1 & 1 \\ -1 & -1 & 2 \\ -1 & -2 & 2 \end{bmatrix}$$

ja

$$\mathbf{1} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T.$$

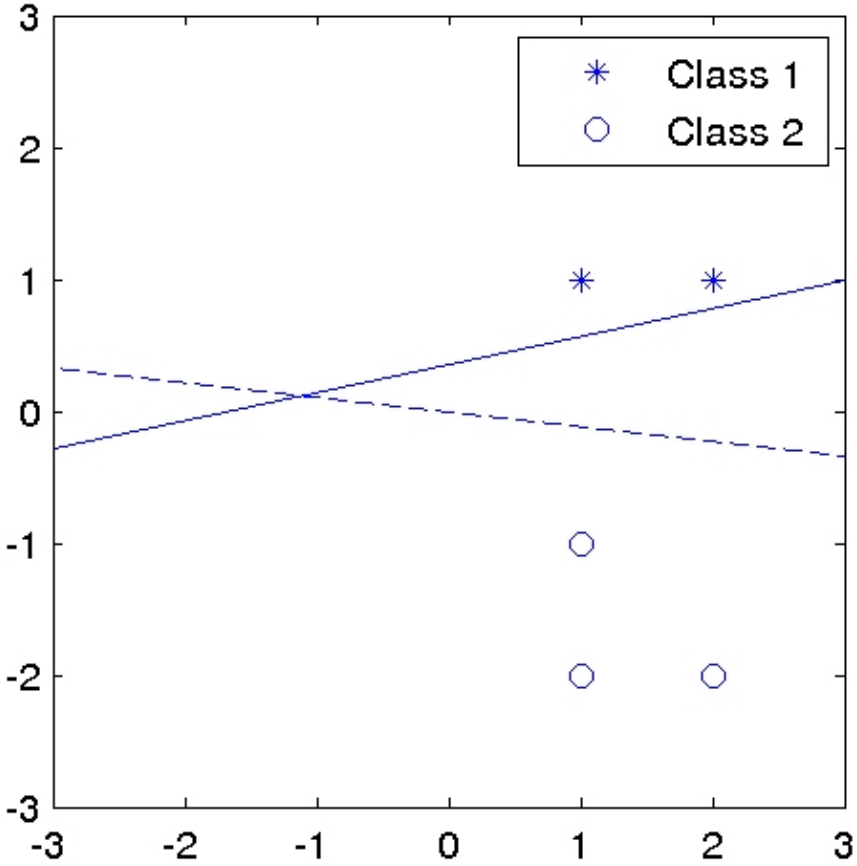
Jatkettu painovektori voidaan laskea esimerkiksi Matlabin avulla

```
>> a = inv(Y'*Y)*Y'*ones(6,1)

a =

    0.0000
    0.0667
    0.6000
```

Täten erotinfunktio on $g(\mathbf{x}) = [0.0667, 0.6]\mathbf{x}$. Päätöspinta voidaan selvittää samoin kuin Perceptronilla opitun luokittimen tapauksessa: pisteet $[0, 0]^T$ ja $[-9, 1]^T$ sijaitsevat päätöspinnalla (ks. kuva 6.3).



Kuva 6.3: Päätöspinnat perceptron kriteerin ja pienimmän neliösumman kriteerin avulla. Harjoitusnäytteet ovat listattu kappaleessa 6.3. Perceptron kriteerin (ja algoritmin) avulla saatua päätöspintaa esittää kiinteä viiva ja pienimmän neliösumman kriteerin avulla saatua päätöspintaa esittää pilkkuviiva.

Luku 7

Luokitinten testaus

7.1 Virhetodennäköisyyden arviointi

Ratkaistessamme luokitusongelmaa on tärkeää estimoida luokittimen virhettä. Tämä tarjoaa vastauksen esimerkiksi kysymyksiin kuten 'onko luokitusjärjestelmä tarpeeksi hyvä vai täytyykö sitä vielä parantaa?' ja 'onko luokitin A parempi kuin luokitin B?'.

Muistutetaan vielä, että olemme kiinnostuneita nimenomaan sen todennäköisyydestä, että ennennäkemätön (siis opetusaineistoon kuulumaton) piirrevektori luokittuu väärin. Tätä kutsuttiin *testivirheeksi* kappaleessa 5.5.

Opetusvirhe on taas virhefrekvenssi opetusaineistolle. Opetusvirhe on huono ja ylioptimistinen estimaatti testivirheelle. Esimerkiksi, NN-luokitinten opetusvirhe on automaattisesti nolla. Samaa ei voida sanoa testivirheestä. Opetusvirheen käytöstä testivirheen estimaattina käytetään nimitystä takaisinsijoitusmenetelmä (engl. resubstitution method).

Takaisinsijoitusmenetelmää huomattavasti parempi arvio testivirheelle saadaan kun jaetaan alkuperäinen opetusdata

$$\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_c$$

kahteen osaan $\mathcal{D}_{opetus}$ ja $\mathcal{D}_{testi}$. Näistä piirrevektorijoukkoa $\mathcal{D}_{opetus}$ käytetään luokittimen opettamiseen ja $\mathcal{D}_{testi}$ on ainoastaan luokitusvirheen suuruuden arviointia varten. Tätä kutsutaan *pidätysmenetelmäksi* (engl. holdout method). Se on parhaimmillaan kun opetusdataa on paljon eikä pieni vähennys opetusnäytteiden määrässä heikennä olennaisesti luokittimen laatua. Tilanteen ollessa tämä pidätysmenetelmä on suositeltava tapa arvioida luokitusvirheen suuruutta. Huomaa, että jaon joukkoihin $\mathcal{D}_{opetus}$ ja $\mathcal{D}_{testi}$ täytyy olla sattumanvarainen. Esimerkiksi ei ole hyvä idea valita joukoksi $\mathcal{D}_{testi}$ luokan 1 harjoitusnäytteitä $\mathcal{D}_1$.

Kun opetusnäytteitä ei ole turhan montaa, täytyy luokitusvirhettä arvioida jonkin ristiinvalidointimenetelmän avulla. Näistä yksinkertaisin on *leave-one-out*-menetelmä. Siinä jätetään joukosta $\mathcal{D}$ vuoronperään kukin opetusnäytteistä pois ja opetetaan luokitin jäljellejääneiden näytteiden avulla. Sitten katsotaan luokittuuko poistettu harjoitusnäyte oikein vai väärin. Kun tämä on toistettu kaikille harjoitusnäytteille,

niin luokitusvirhettä voidaan arvioida

$$\hat{E}(\alpha) = \frac{v}{n},$$

missä v on väärin luokittuneiden harjoituspiirrevektoreiden lukumäärä ja $n = \sum_{i=1}^{c} n_i$ on harjoitusnäytteiden kokonaismäärä. (Hattunotaatio tarkoittaa tässäkin estimaattia, ts. $\hat{E}(\alpha)$ on estimaatti todellisesta luokitusvirheestä $E(\alpha)$.) Menetelmä tuottaa hyvän arvion luokitusvirheelle. Menetelmän haittapuolena on laskennallinen vaativuus, täytyyhän nyt suunnitella n luokitinta yhden sijasta.

7.2 Sekaannusmatriisi

Kun arvioidaan luokitusvirheen kokoa on usein hyödyllistä tarkastella myös *sekaannusmatriisia* (engl. confusion matrix). Sekaannusmatriisin

$$S = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1c} \\ \vdots & & \ddots & \vdots \\ s_{c1} & s_{c2} & \cdots & s_{cc} \end{bmatrix}$$

alkio s_{ij} kertoo niiden testinäytteiden lukumäärän, jotka tulevat luokitetuksi luokkaan ω_i, mutta joiden todellinen luokka on ω_j. Indeksien järjestys voitaisiin tietenkin valita myös toisin ja se vaihtelee lähdeteoksesta riippuen, josta syystä on hyvä olla tarkkana eri lähdeteoksia vertailtaessa.

Luokitusvirhe saadaan sekaannusmatriisista vähentämällä diagonaalialkioiden summa alkioiden kokonaissummasta ja jakamalla tämä erotus alkioiden kokonaissummalla. Toisin sanoen:

$$\hat{E}(\alpha) = \frac{\sum_{i=1}^{c}\sum_{j=1}^{c} s_{ij} - \sum_{i=1}^{c} s_{ii}}{\sum_{i=1}^{c}\sum_{j=1}^{c} s_{ij}}.$$

Sekaannusmatriisin perusteella voidaan myös laskea useita muita luokittimen suunnittelussa tärkeitä suureita.

7.3 Esimerkki

Tarkastelemme vielä leave-one-out virheenarviointimenetelmää Fisherin Iirisdatan tapauksessa. Tutkimme nimenomaisesti ML-estimaatteihin perustuvaa luokitusta ja oletamme, että luokat ovat normaalijakautuneita. Data oli kerätty erillisen otannan avulla, joten oletamme prioritodennäköisyydet yhtä suuriksi. Matlabiin tämä aineisto voidaan ladata

```
>> load fisheriris
```

komennon avulla. (Ainakin, jos statistics toolbox on saatavissa). Nyt Matlabissa pitäisi olla muuttujat 'meas', joka sisältää piirrevektorit 150:sta iiriksestä ja 'species'

joka paljastaa kutakin piirrevektoria vastaavan luokan. Huomataan, että piirrevektorit 1 - 50 kuuluvat luokkaan ω_1 (iris setosa), piirrevektorit 51 - 100 luokkaan ω_2 (iris versicolor) ja piirrevektorit luokkaan ω_3 (iris virginica). Tämä helpottaa toteutusta Matlabissa. Seuraava Matlab-koodi hoitaa virheenarvioinnin (itseasiassa laskemme sekaannusmatriisin):

```
% alustetaan sekaannusmatriisi
s = zeros(3,3);
% luokka 1, piirrevektorit 1 - 50
% luokka 2, piirrevektorit 51 - 100
% luokka 3, piirrevektorit 101 - 150
for i = 1:150
  trueclass = ceil(i/50);
  test = meas(i,:); % testivektori on i:s piirrevektori
  % luokalle 'trueclass opetusdatana ovat kaikki muut piirrevektorit paitsi
  % testivektori, muille luokille ovat käytössä kaikki piirrevektorit
  train1 = [meas(1:min((i - 1),50),:);meas((i + 1):50,:)];
  train2 = [meas(51:min(i - 1,100),:);meas(max((i + 1),51):100,:)];
  train3 = [meas(101:(i - 1),:);meas(max(101,(i + 1)):150,:)];
   % Opetetaan luokitin, ts. lasketaan luokkien keskiarvovektorit ja
  % kovarianssimatriisit opetusdatan perusteella
  mu1 = mean(train1);
  cov1 = cov(train1);
  mu2 = mean(train2);
  cov2 = cov(train2);
  mu3 = mean(train3);
  cov3 = cov(train3);
  % luokitin on opetettu, lasketaan posterior todennäköisyydet
  % ja  katsotaan mihin luokkaan testipiste
  % luokittuu
  posterior(1) = mvnpdf(test,mu1,cov1);
  posterior(2) = mvnpdf(test,mu2,cov2);
  posterior(3) = mvnpdf(test,mu3,cov3);
  [maxposterior,testclass] = max(posterior);
  % ja kasvatetaan vastaavaa sekaannusmatriisin alkiota
  % yhdellä
  s(testclass,trueclass) = s(testclass,trueclass) + 1;
end
```

Yllä olevassa koodissa funktio 'ceil' palauttaa reaaliluvulle lähimmän sitä suuremman kokonaisluvun. Koodissa myös käytetään hyväksi sitä, että esimerkiksi `meas(56:50,:)` on tyhjä matriisi.

Sekaannusmatriisiksi tulee

```
s =
```

```
50  0  0
 0 47  1
 0  3 49
```

Josta voimme laskea arvion luokitusvirheelle

$$\hat{E}(\alpha) = 4/150 \approx 0.027.$$

Luku 8

Ohjaamaton luokitus eli klusterointi

8.1 Johdanto

Tähän asti olemme suunnitelleet luokittimia perustuen olettamukseen, että meillä on käytössämme joukko opetusnäytteitä joiden oikea luokka tunnetaan. Toisin sanoen olemme tutustuneet ohjattuun luokitukseen (ks. kappale 2.4). Joskus luokitinten suunnittelussa täytyy kuitenkin turvautua ohjaamattomaan luokitukseen, eli klusterointiin[1]. Klusteroinnissa meillä ei ole käytössä harjoitusnäytteitä vaan luokitin täytyy 'opettaa' ilman tietoa piirrevektorien oikeista luokista.

Selvästikin klusterointi on kertaluokkaa ohjattua luokitusta vaikeampi ongelma. Tilanteita, joissa klusterointi on välttämätöntä tai hyödyllistä, ovat esimerkiksi:

1. Opetusnäytteiden kerääminen ja luokitus ovat usein erittäin aikaa vieviä tehtäviä, ja siksi harjoitusnäytejoukon kerääminen voi olla mahdotonta. Myöskin, jos opetusnäytteitä on paljon, on mahdollista, että näistä kaikkia ei pystytä käsin luokittamaan. Tällöin voi olla hyödyllistä opettaa luokitin pienellä joukolla näytteitä ja sen jälkeen hienosäätää sitä luokittamattomien piirrevektorien perusteella.

2. Toisen suuntainen tilanne tulee esiin tiedon louhinnassa, missä on edullista etsiä 'luonnollisia' ryhmiä (klustereita) piirrevektoreiden joukosta ja sen jälkeen tunnistaa ryhmät.

3. Piirrevektoreiden ominaisuudet voivat muuttua rajustikin ajan kuluessa. Tällöin ohjattu luokitus ei ole järkevää, koska testi-piirrevektorit olisivat ennen pitkää täysin erilailla jakautuneita kuin opetukseen käytetyt piirrevektorit. Esimerkiksi lääketieteellisten kuvien käsittelyssä tämä ongelma on yleinen.

4. Klusterointia voidaan myös hyödyntää ohjatussa luokituksessa, esimerkiksi etsittäessä sopivia parametrisia muotoja luokkatiheysfunktioille.

[1]Kielitoimiston termi klusteroinnille on ryvästys, mutta mielestäni klusterointi on terminä tarpeeksi suomenkielinen

Yllä-olevien tilanteiden lisäksi ohjaamaton luokitus on hyödyllistä useista muistakin syistä.

8.2 Klusterointiongelma

Jäsentelemme nyt klusterointiongelman täsmällisemmin. Annettuna on joukko piirrevektoreita

$$\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}.$$

Tehtävänä on sijoittaa nämä vektorit johonkin c:stä luokasta. Toisin sanoen tehtävänä on löytää sellaiset c joukkoa $\mathcal{D}_1, \ldots, \mathcal{D}_c$, joille

$$\bigcup_{i=1}^{c} \mathcal{D}_i = \mathcal{D}$$

ja

$$\mathcal{D}_j \cap \mathcal{D}_i = \emptyset$$

kaikilla $i \neq j$.

Luonnollisesti, jotta ongelma olisi mielekäs, tarvitaan joitakin lisäoletuksia. (Mielivaltainen piirrevektorien jako eri luokkiin ei ole kovin hyödyllinen.) Tällä kurssilla oletamme, että voimme mitata piirrevektorien samankaltaisuutta jollakin tavalla. Siten tehtäväksi muodostuu kunkin luokan piirrevektorien samankaltaisuuden maksimointi. Tätä varten meidän täytyy pystyä mittaamaan kuinka samankaltaisia johonkin joukkoon $\mathcal{D}_i$ kuuluvat piirrevektorit ovat.

Seuraavissa kappaleissa esittelemme kaksi eri vaihtoehtoa piirrevektoreiden samankaltaisuuden määrittelyyn. Olemme jo tutustuneet molempiin ohjatun luokituksen yhteydessä. Ensimmäinen tavoista on minimietäisyysluokittimen sovellus klusterointiin. Tämä johtaa *k-means* algoritmiin. Toinen tavoista perustuu suurimman uskottavuuden estimaattiin ja niin sanottuihin *sekoitemalleihin*.

8.3 K-means klusterointi

8.3.1 K-means kriteeri

Palautetaan mieleen, että minimietäisyysluokitin luokitti piirrevektorin siihen luokkaan, jonka keskiarvovektori oli lähinnä piirrevektoria. Merkitään nyt luokan $\mathcal{D}_i$ alkioiden määrää symbolilla n_i. (Tätä siis ei vielä tunneta).

Ongelmana on, ettei keskiarvovektoreita tunneta, mutta määritellään siitä huolimatta luokan $\mathcal{D}_i$ samankaltaisuus

$$s(\mathcal{D}_i) = -\sum_{\mathbf{x} \in \mathcal{D}_i} ||\mathbf{x} - \mu_i||^2,$$

missä $\mu_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}$. Ratkaistaan nyt keskiarvovektorit maksimoimalla luokkien yhteenlaskettu samankaltaisuus. Yhtälailla voimme minimoida samankaltaisuuden

vastalukua:

$$J(\mathcal{D}_1, \ldots, \mathcal{D}_c) = \sum_{i=1}^{c} \sum_{\mathbf{x} \in \mathcal{D}_i} ||\mathbf{x} - \mu_i||^2. \tag{8.1}$$

Tämä on *k-means kriteeri*.

8.3.2 K-means algoritmi

Huomaa, että tuntemalla klusterit $\mathcal{D}_1, \ldots, \mathcal{D}_c$ voimme laskea keskiarvovektorit ja tuntemalla keskiarvovektorit voimme laskea klusterit. Tämä havainto on tärkeä funktion (8.1) minimoimiseksi. (Koska J on joukon $\mathcal{D}$ partitioiden funktio niin esimerkiksi luvussa 6 käytetty gradienttimenetelmä ei kelpaa). Algoritmia funktion (8.1) minimoimiseksi kutsutaan *k-means algoritmiksi*. Perusidea on laskea keskiarvovektorit luokituksen perusteella, luokittaa piirrevektorit uudelleen uusien keskiarvovektorien perusteella ja toistaa tätä kunnes luokitus ei enää muutu.

---

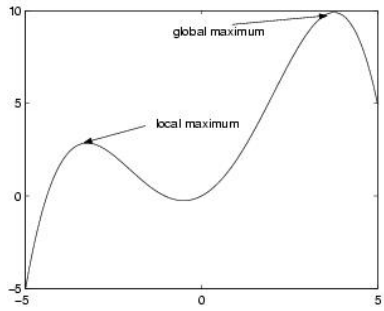
k-means algoritmi
Alusta $\mu_1(0), \ldots, \mu_c(0)$, aseta $t \leftarrow 0$
repeat
  Luokita pisteet $\mathbf{x}_1, \ldots, \mathbf{x}_n$ luokkiin $\mathcal{D}_j(t)$, joiden keskiarvovektori $\mu_j(t)$ lähinnä.
  for $k = 1$ to c **do**
    laske uudet keskiarvovektorit $\mu_k(t+1) = \frac{1}{|\mathcal{D}_k(t)|} \sum_{\mathbf{x} \in \mathcal{D}_k(t)} \mathbf{x}$
  end for
  Aseta $t \leftarrow t + 1$
until Luokitus ei enää muutu
Palauta $\mathcal{D}_1(t-1), \ldots, \mathcal{D}_c(t-1)$.

Huomautuksia: 1) $\mathcal{D}_j(t-1)$ palautetaan koska iteraatiolaskuria ehdittiin kasvattaa yhdellä juuri ennen algoritmin päättymistä. 2) Merkintä $|\mathcal{D}_k(t)|$ tarkoittaa joukon $\mathcal{D}_k(t)$ alkioiden lukumäärää.

Tyypillisesti keskiarvovektorit alustetaan satunnaisesti valittujen piirrevektoreiden arvoilla. Esimerkiksi, yksinkertainen tapa alustaa on asettaa $\mu_j(0) \leftarrow \mathbf{x}_j$. Tyypillisesti myös algoritmin tulos riippuu annetusta alustuksesta. Tämä johtuu siitä, että k-means algoritmi on lokaalinoptimoinnin algoritmi ja minimointiongelmalla on tässä tapauksessa useita lokaaleja minimejä (ks. kuva 8.1). K-meansin tapauksessa näitä ei kuitenkaan ole häiritsevän paljoa.

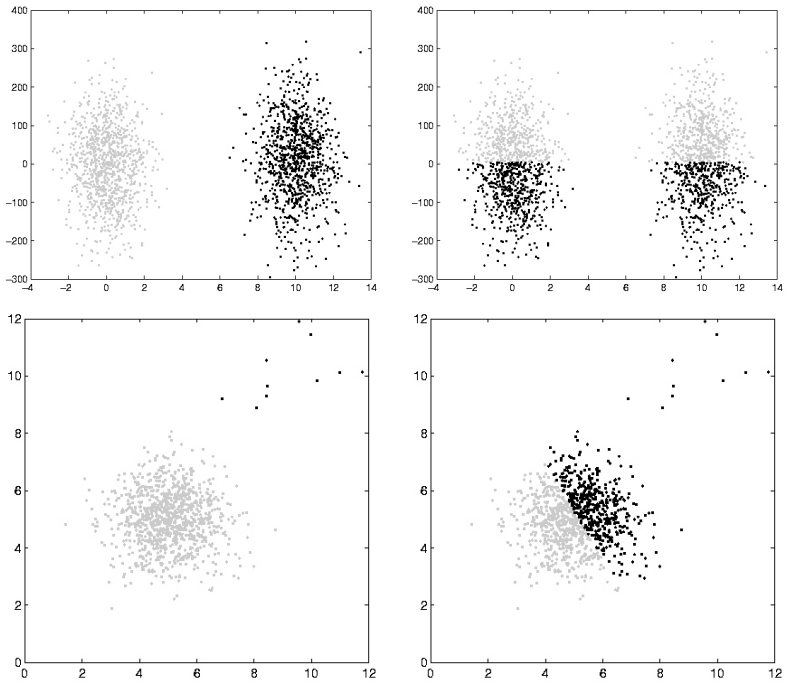
8.3.3 K-means klusteroinnin ominaisuuksia

K-means klusterointi on minimietäisyysluokittimen 'sovellus' klusterointiin. Siispä k-means klusterointi kärsii samoista ongelmista kuin minimietäisyysluokitin: Jos klusterit ovat eri kokoisia (sisältävät eri määrän piirrevektoreita), niin k-means ei löydä hyvää klusterointia, katso kuva 8.2. Sama ongelma toistuu, jos klustereilla on kovin erilaiset kovarianssit tai piirteiden luokkakohtaisissa variansseissa on suuria



Kuva 8.1: Lokaali vs. globaali maksimi

eroja. Näiden klusterointiongelmien ratkaisuun tarvitsemme siis monimutkaisemman algoritmin. Sen sijaan, jos klusterit ovat saman suuruisia ja ne ovat 'tiukkoja', k-means pärjää erinomaisesti.



Kuva 8.2: K-means algoritmille ongelmallisia tilanteita. Ylärivi: Piirteillä erilaisen skaala (huomaa akseleiden sklaalaus). Alarivi: Toisesta luokasta huomasttavasti enemmän edustajia. 'Todelliset luokat' näytetty vasemmalla ja oikealla on klusterointitulos k-means algoritmilla.

8.4 Sekoitemallit

Tarkastelemme nyt normaalijakaumiin [2] perustuvia sekoitemalleja. Tällöin yritämme ohjaamattomasti oppia Bayes luokittimen kun luokkien tiheysfunktioiden parametriset muodot tunnetaan. Oletamme siis seuraavaa:

1. Luokkatiheysfunktiot ovat normaalijakauteneita $p(\mathbf{x}|\omega_j) = p_{normal}(\mathbf{x}|\mu_J, \Sigma_j)$ kaikille $j = 1, \ldots, c$.

2. parametrit $\mu_1, \ldots, \mu_c, \Sigma_1, \ldots, \Sigma_c$ ovat tuntemattomia.

3. $P(\omega_1), \ldots, P(\omega_c)$ ovat tuntemattomia

4. Opetusdataa ei ole käytettävissä. Siis piirrevektorien $\mathbf{x}_1, \ldots, \mathbf{x}_n$ luokat ovat tuntemattomia ja tehtävänä on selvittää nuo luokat.

Oletamme, että piirrevektorit ovat riippumattomien satunnaismuuttujien realisaatioita ja että kukin niistä on jakautunut jonkin luokkatiheysfunktion mukaan. Mutta, kuten todettua, emme tiedä minkä. Voimme kuitenkin kirjoittaa todennäköisyystiheyden sille, että piirrevektori $\mathbf{x}_i$ havaitaan ja se kuuluu johonkin luokista. Ensinnäkin

$$p(\mathbf{x}_i, \omega_j) = P(\omega_j)p(\mathbf{x}_i|\omega_j)$$

kaikille i ja j. Tämä ei kuitenkaan vielä auta paljoa, sillä emme tiedä minkä luokan tiheyden mukaisesti $\mathbf{x}_i$ on jakautunut. Mutta nyt

$$p(\mathbf{x}_i) = \sum_{j=1}^{c} p(\mathbf{x}_i, \omega_j),$$

sillä $\mathbf{x}_i$:n on kuuluttava johonkin luokista $\omega_1, \ldots, \omega_c$. (Ylläolevassa kaavassa vasen puoli on marginaalitiheys.) Nämä yhdistämällä

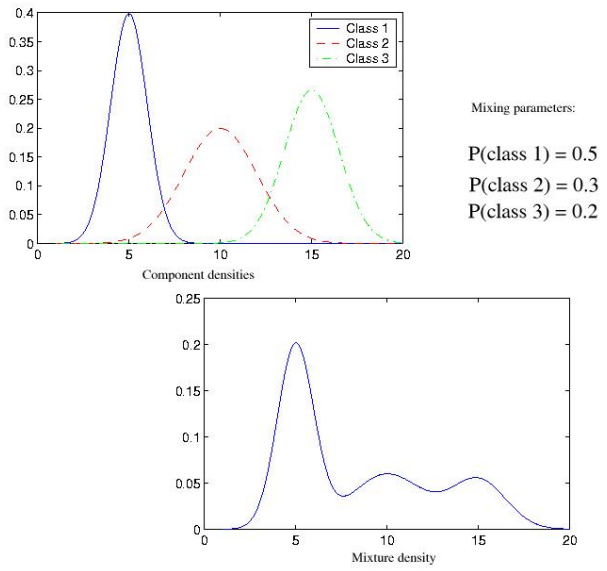
$$p(\mathbf{x}_i) = \sum_{j=1}^{c} p(\mathbf{x}_i|\omega_j)P(\omega_j). \tag{8.2}$$

Tätä kutsutaan *sekoitetiheydeksi* (engl. mixture density). Sekoitetiheyksien prioreita $P(\omega_i)$ kutsutaan *sekoitusparametreiksi* (engl. mixing parameters). Luokkatiheysfunktioita kutsutaan *komponenttitiheyksiksi*.

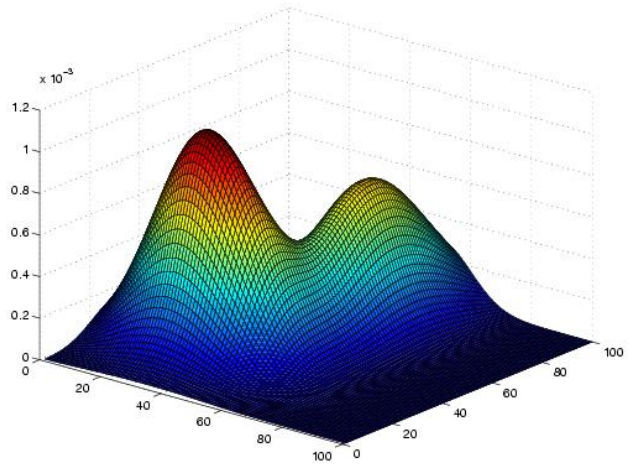
Oletimme, että komponenttitiheydet ovat normaalijakautuneita, joten saamme parametrisen sekoitetiheyden

$$p(\mathbf{x}_i|\theta) = \sum_{j=1}^{c} p_{normal}(\mathbf{x}_i|\mu_j, \Sigma_j)P(\omega_j), \tag{8.3}$$

[2] Sekoitemalleja voidaan johtaa muunkin tyyppisille jakaumille, mutta tällä kurssilla rajoitumme normaalijakaumien tarkasteluun.



Kuva 8.3: Sekoitemallit. Ylhäällä komponenttitiheydet ja alhaalla tuloksena oleva sekoitetiheys kun sekoitusparametrit ovat kuten kuvassa.



Kuva 8.4: Sekoitemalli kahden piirteen tapauksessa

jossa parametrivektori

$$\theta = (\mu_1, \ldots, \mu_c, \Sigma_1, \ldots, \Sigma_c, P(\omega_1), \ldots, P(\omega_c))^T.$$

Tämän perusteella voimme johtaa algoritmin parametrien ML-estimaatin $\hat{\theta}$ ratkaisemiseksi. Siitä seuraavassa kappaleessa. Menemme hiukan asioiden edelle ja katsomme miten piirrevektori $\mathbf{x}_i$ voidaan luokittaa, kun meillä on käytössämme ML-estimaatit kaikista vaadituista parametreista. Kuten arvata saattaa, käytämme Bayes luokitinta johon vain sijoitamme estimoidut parametrit todellisten tilalle. Eli:

$$\alpha_{mixture}(\mathbf{x}_i) = \arg \max_{\omega_j, j=1,\ldots,c} p(\mathbf{x}_i|\hat{\mu}_j, \hat{\Sigma}_j)\hat{P}(\omega_j).$$

Sekoitemallien ongelmana on, että ne eivät aina ole tunnistettavissa (engl. identifiable). Tämä tarkoittaa, että kaksi eri parametrivektoria θ_1, θ_2 saattavat tuottaa täsmälleen saman tiheysfunktion. Eli

$$p(\mathbf{x}|\theta_1) = p(\mathbf{x}|\theta_2)$$

kaikilla $\mathbf{x}$. Tämä on varsin suuri ongelma. Tunnistettavuusongelman erikoistapaus on *vaihto-ongelma* (engl. switching problem): Jos vaihdamme luokat ω_i ja ω_j päikseen niin saamme saman sekoitetiheyden (kunhan komponenttitiheyksillä on sama parametrinen muoto). Luokat siis täytyy nimetä jälkikäteen. Tämä on joskus vaikea ongelma, jos todella halutaan suunnitella ohjaamaton luokitin.

8.5 EM algoritmi

Palaamme nyt sekoitetiheyden parametrien ratkaisemiseen. Koska piirrevektorit $\mathbf{x}_1, \ldots, \mathbf{x}_n$ olivat riippumattomia niin voimme kirjoittaa uskottavuusfunktion

$$\prod_{i=1}^{n} p(\mathbf{x}_i|\theta) = \prod_{i=1}^{n} \sum_{j=1}^{c} p_{normal}(\mathbf{x}_i|\mu_j, \Sigma_j)P(\omega_j).$$

Maksimoimalla tämän parametrivektorin θ suhteen saamme ML-estimaatin $\hat{\theta}$. Jälleen on mukavampi toimia uskottavuusfunktion logaritmin avulla, eli

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \ln[\sum_{j=1}^{c} p_{normal}(\mathbf{x}_i|\mu_j, \Sigma_j)P(\omega_j)]. \tag{8.4}$$

Tämä on valitettavan vaikea funktio maksimoida. (Itse asiassa olemme kiinnostuneita tässä suurimmasta lokaalista maksimista, koska uskottavuusfunktio ei ole rajoitettu, katso esimerkiksi Duda, Hart, Stork luku 10). Mutta EM (expectation maximization) algoritmi sen lokaaliin maksimointiin voidaan aika helpostikin johtaa. Jätämme johdon tällä kurssilla kuitenkin väliin, mutta annamme itse algoritmin.

EM-algoritmi

1. Alusta $\mu_j^0, \Sigma_j^0, P^0(\omega_j)$, aseta $t \leftarrow 0$.

2. (E-askel) Laske posterior todennäköisyydet, että $\mathbf{x}_i$ kuuluu luokkaan ω_j

$$p_{ij}^{t+1} = \frac{P^t(\omega_j) p_{normal}(\mathbf{x}_i | \mu_j^t, \Sigma_j^t)}{\sum_{k=1}^{c} P^t(\omega_k) p_{normal}(\mathbf{x}_i | \mu_k^t, \Sigma_k^t)}.$$

3. (M-askel) Laske uudet parametrien arvot

$$P^{t+1}(\omega_j) = (1/n) \sum_i p_{ij}^{t+1} \tag{8.5}$$

$$\mu_j^{t+1} = \frac{\sum_i p_{ij}^{t+1} \mathbf{x}_i}{\sum_i p_{ij}^{t+1}} \tag{8.6}$$

$$\Sigma_j^{t+1} = \frac{\sum_i p_{ij}^{t+1} (\mathbf{x}_i - \mu_j^{t+1})(\mathbf{x}_i - \mu_j^{t+1})^T}{\sum_i p_{ij}^{t+1}} \tag{8.7}$$

4. Aseta $t \leftarrow t + 1$.

5. Lopeta jos jokin kovergenssiehto on täyttynyt, muutoin palaa kohtaan 2.

EM-algoritmi löytää yleensä uskottavuusfunktion lokaalin maksimin. Nyt yleensä kuitenkin lokaaleja maksimeja on useita, joten lopputulos riippuu alustuksesta.