UPTEC X09 026
Examensarbete 30 hp
Juni 2009

Prestandautvärdering vid multivariat klassning av blodprover för cancer-detektion

Christofer Bäcklin

UPPSALA UNIVERSITET

Bioinformatics Engineering Program

Uppsala University School of Engineering

UPTEC X 09 026	Date of issue 2009-06

Author

Christofer Bäcklin

Title (English)

Performance estimation of multivariate classification of blood samples for cancer detection

Title (Swedish)

Prestandautvärdering vid multivariat klassning av blodprover för cancerdetektion

Abstract

Disease detection methods involving multivariate classification need reliable performance estimates to be practically useful. A new method for making such estimates is presented here based on splitting the data in several bags and estimating the error rate by training many classifiers on new data drawn from kernel density estimations of the unknown underlying distribution in each bag. This gives an estimation of the entire prior function of the classification problem together with an uncertainty measure which is a great improvement from the standard methods of today which only supply its mean and no uncertainty measure.

Keywords

Classification performance, error rate, kernel estimation, cancer detection, in solution PLA

Supervisors

Claes Andersson
Uppsala university, department of medical sciences

Scientific reviewer

Mats Gustafsson
Uppsala universitet, the Linnaeus center for bioinformatics

Project name	Sponsors
Language **Swedish**	Security **Secret until 18th June 2011**
ISSN 1401-2138	Classification
Supplementary bibliographical information	Pages **27**

Biology Education Centre Biomedical Center Husargatan 3 Uppsala
Box 592 S-75124 Uppsala Tel +46 (0)18 4710000 Fax +46 (0)18 555217

Prestandautvärdering vid multivariat klassning av blodprover för cancerdetektion

Christofer Bäcklin

16 juni 2009

Sammanfattning

Här presenteras en alternativ metod för att utvärdera prestandan hos klassificerares designprocedurer som är mer informativ än dagens mest populära tekniker, korsvalidering (CV) och bootstrap (BTS). Vår metod skattar problemets hela fördelning av felsannolikheter som klassificerare byggda m.a.p. det givna problemet kan anta (dess prior) samt osäkerheten i skattningen medan CV och BTS endast skattar dess medel.

Metoden går ut på att först dela upp all data i separata delmängder. I varje del uppskattas den sanna fördelningen med en bayesiansk kärnskattning utifrån vilken det tränas och testas en stor mängd klassificerare. Klassificerarnas testresultat kan sedan användas för att skatta problemets prior. Metodens skattade prior konvergerar mot den sanna med ökande mängd data men ofta krävs det tusentals observationer innan man får en acceptabel noggrannhet ens i två dimensioner.

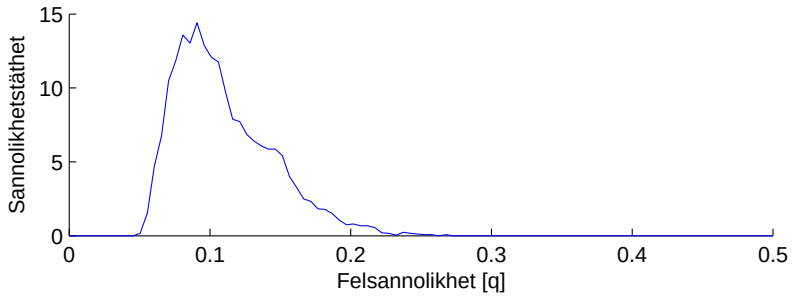
Arbetet ingår i EU-projektet ProActive vars mål är att undersöka om det går att upptäcka cancer i tidiga skeden genom att mäta koncentrationen av olika markörproteiner i blodprover. Utifrån koncentrationerna vill man ställa diagnoser m.h.a. en klassificerare och utvärdera designproceduren med den nya metoden.

Examensarbete 30hp
Civilingenjör i bioinformatik
Uppsala Universitet

Innehåll

1 Introduktion

Klassificeringsproblem där man utifrån ett antal kända observationer vill gissa klasstillhörighet på nya okända observationer är idag vanliga. Det kan t.ex. handla om att få en maskin att känna igen handskrivna tecken, sortera ut skadade produkter på ett löpande band eller avgöra om en patient är sjuk utifrån testresultat. För att en klassificerare ska vara användbar i praktiken måste den levereras med ett prestandamått och i de fall mängden kända observationer är begränsad måste konstruktören kompromissa mellan hur många som ska användas för att träna och testa klassificeraren. Då är det naturligtvis intressant att få ut så mycket information som möjligt ur sina observationer och för det har det föreslagits olika metoder som bygger på att återanvända dem upprepade gånger på kontrollerade sätt. De mest populära metoderna för det idag är korsvaildering (CV) och bootstrap (BTS) [11]. I och med att de återanvänder datamängden upprepade gånger skattar de dock inte prestandan hos en individuell klassificerare utan istället skattar de hur bra hela proceduren som bildar sådana klassificerare är, den s.k. *designproceduren* (se del 2), och då endast dess medel. CV och BTS säger alltså inget om designprocedurens spridning i prestanda vilket betyder att om medelprestandan uppskattas till en felsannolikhet på 25% har vi ingen aning om alla individuella klassificerare den ger upphov till ligger i närheten av 25% eller om vissa i princip är perfekta och andra singlar slant om beslutet. I det här arbetet presenteras därför en alternativ metod för att skatta designprocedurers prestanda i formen av dess prior-funktion samt ett mått på osäkerheten i skattningen. Priorn är sannolikhetstätheten över hur troligt det är att få klassificerare av en viss felsannolikhet på en fördelning, se exemplet i fig. 1.



Figur 1: Ett exempel på en designprocedurs prior-funktion. De flesta klassificerare den tränar har en felsannolikhet på c:a 11% men det finns utstickare som går ända ner till 5% eller upp till 25%.

Resten av del 1 innehåller en presentation av projektet som arbetet ingår i och en crash course i bayesiansk interferens. Del 2 redogör för hur man idag uppskattar prestanda hos klassificerare och designprocedurer och del 3 problematiserar kort över bristerna med det och den allmänna okunskapen om hur metoderna ska användas. Vår nya metod introduceras i del 4 och resultatet av de tester den genomgått redovisas i del 5. Därefter följer en diskussion kring resultatet i del 6 och en kort plan för framtida utveckling i del 7.

1.1 ProActive-projektet

Det här examensarbetet ingår i Uppsala universitets bidrag till EU-projektet ProActive som leds av företaget Olink Bioscience AB i Uppsala. Målet med projektet är att utveckla metoder för att upptäcka kolorektalcancer i tidiga stadier och avgöra vilken typ det rör sig om så att man kan agera innan sjukdomen fått fäste och sätta in rätt behandling (därav projektets namn som syftar på att agera i förtid i motsats till reaktiva behandlingar). Tidigare har cancerforskning koncentrerats på att ta fram metoder för att behandla cancer i sena stadier så det här är en relativt ny infallsvinkel.

I projektet ingår följande medlemmar.

- Olink Bioscience, Uppsala
- Uppsala universitet, inst. för medicinska vetenskaper
- Fujirebio Diagnostics AB, Göteborg
- Innova Biosciences Ltd, Cambridge (UK)
- Integromics S.L., Granada (ES)
- Köpenhamns universitet (DK)

Olink driver projektet och utvecklar tekniken för att mäta biomarkörerna med hjälp av Fujirebio och Innova. Köpenhamns universitet tillhandahåller patientproverna och Integromics står för informationshanteringssystemet (LIMS) och visualiseringsverktyg. Uppsala universitets del i projektet går ut på ta fram bra detektorer (d.v.s. klassificerare) och det här examensarbetet går i sin tur ut på att utveckla och implementera standard operating procedures (SOP) för att ta reda på hur bra designproceduren är. Det här arbetet har alltså inget att göra med hur bra en enskild klassificerare är eller säkerheten i dess bedömningar utan går ut på att utvärdera hur bra klassificerare man kan förvänta sig att få med den designprocedur som används på den data som finns tillgänglig.

Mer om projektet finns att läsa på
`http://www.olink.se/proactive/proactive_content.php`.

1.2 Datamängd

I projektet kommer blodprover från 800-1000 patienter att analyseras med s.k. *in solution proximity ligation assay* (PLA) där man mäter halterna av 180 markörproteiner. Markörproteinerna är sådana som tros ha en koppling till cancer och har valts ut efter en litteraturstudie. Mer om PLA och hur tekniken tillämpas för cancerdetektion går att läsa i Fredriksson et al. [5, 6, 7] och Gullberg et al. [8].

I det här arbetet har enbart konstgjord data använts för att studera metoden eftersom den kan framställas i obegränsad mängd.

1.3 Kort om bayesiansk inferens

Vår metod grundar sig till stor del på bayesiansk interferens [16] och för att förstå innehållet i rapporten krävs det att man känner till dess grundläggande termer och principer. I centrum står Bayes sats, ekv. 1, som anger sannolikheten att en hypotes H är sann givet viss information eller data D. Nedanför finns förklaringar av de olika termerna tillsammans med ett exempel för att illustrera dem.

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)} \tag{1}$$

- $P(H|D)$ är sannolikheten att hypotesen stämmer givet datamängd D vilket är det man oftast är mest intresserad av. Den kallas för *a posteriori-sannolikheten* eller *posteriorn*.
- $P(H)$ är sannolikheten att hypotesen stämmer i ett godtyckligt fall d.v.s. före man sett någon data. Den kallas för *a priori-sannolikheten* eller *priorn* och är i regel betydligt svårare att räkna ut än posteriorn.
- $P(D)$ är sannolikheten att man får just den data man fått (oavsett hypotes). Den är i regel också svår att räkna ut och används ofta som en normaliseringsfaktor om man testar många hypoteser på samma datamängd.
- $P(D|H)$ är sannolikheten att observera D givet hypotesen, d.v.s. förutsatt att hypotesen stämmer hur troligt det är att observera den data vi fått. Den kallas för *likelihood*.

Exempel: Du står vid en väg, ser en lastbil passera och vill räkna ut sannolikheten att föraren är en kvinna. Då är din hypotes att föraren är en kvinna och datamängden kan t.ex. innehålla bilens typ, d.v.s. att bilen är en lastbil[1].

- Priorn $P(H)$ anger sannolikheten att en godtycklig bilförare är en kvinna vilken kan uppskattas om man känner till hur ofta män och kvinnor kör bil. Antag att män och kvinnor kör lika mycket, $P(Kvinna) = 0.5$.
- $P(D)$ anger hur vanlig den observerade biltypen är. Antag att lastbilar står för 10% av den totala biltrafiken, $P(Lastbil) = 0.1$.
- $P(D|H)$ anger hur vanlig den observerade biltypen är bland kvinnor. Antag att lastbilar endast står 2% av kvinnors totala bilkörning, $P(Lastbil|Kvinna) = 0.02$.
- Posteriorn anger sannolikheten att föraren av den observerade bilen är en kvinna. Enligt ovanstående antaganden kan man räkna att sannolikheten att en kvinna kör lastbilen är 10%.

$$P(Kvinna|Lastbil) = \frac{P(Lastbil|Kvinna)P(Kvinna)}{P(Lastbil)} = \frac{0.02 \cdot 0.5}{0.1} = 0.1$$

[1] Ofta är det inte självklart vad som ska ingå i datamängden. I den här situationen skulle man t.ex. kunna tänka sig att även tidpunkten och platsen spelar in. Valet är upp till den som samlar in datan. Eventuell data som inte beror av hypotesen faktoriseras automatiskt bort av Bayesmaskineriet.

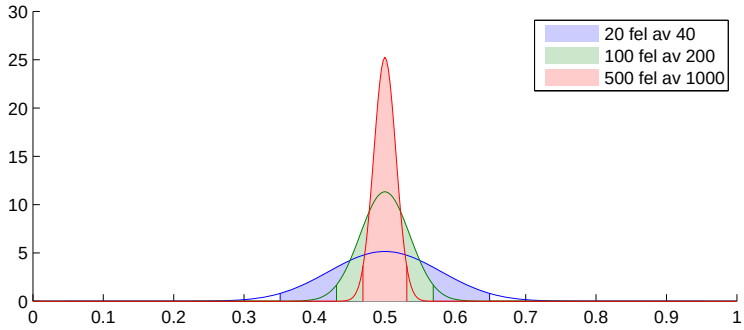
2 Uppskattning av prestanda

När man löser ett problem med lärande system är man naturligtvis intresserad av att veta vilken prestanda man fått, d.v.s. hur ofta gissar den resulterande klassificeraren rätt när den tilldelar klass på en ny okänd observation. Att noggrant uppskatta prestandan hos en enskild klassificerare kräver dock en stor testmängd och det visar sig ofta mer lönsamt att istället uppskatta prestandan hos den designprocedur som skapat klassificeraren.

2.1 Prestandan hos en klassificerare

Vanligtvis använder man en klassificerares felsannolikhet q som prestandamått vilken man kan uppskatta med ett *hold-out-test* som går ut på att testa den på ett antal kända observationer som ej använts för träning och se hur ofta den klassar dem korrekt. Den förväntade prestanda kan sedan räknas ut som en posterior-funktion där hypotesen är att klassificeraren har en viss felsannolikhet och datamängden består av testresultatet, inte den ursprungliga datamängden den tränats eller testats på. Se ekv. 2 och fig. 2 där N_t är antalet testobjekt och k_t är antalet som klassats felaktigt. Priorn $p(q)$ kan vi inte uttala oss om och sätts därför till en konstant över alla felsannolikheter eftersom det är det enda objektiva sättet att välja den på (Laplaces invariansprincip [12]). Den smälter därmed samman med normaliseringskonstanden $p(k_t|N_t)$ och försvinner ur ekvationen.

$$P(q|k_t, N_t) = \frac{p(k_t|q, N_t)p(q)}{p(k_t|N_t)} = \frac{q_t^k(1-q)^{N_t-k_t}}{\int_{q=0}^{1} q_t^k(1-q)^{N_t-k_t}dq} \tag{2}$$



Figur 2: Täthetsfunktion för a posteriori-sannolikheten. De färgade områdena visar 95% av ytan under graferna.

Som det syns i fig. 2 krävs det mycket träningsdata för att få ett snävt intervall i vilket man kan vara säker på att den sanna felsannolikheten befinner sig. Nöjer man sig inte med ett intervall som med 95% sannolikhet täcker det sanna värdet utan man kanske vill ha 99% eller rent av 99,9% krävs det ännu mer data. I verkliga tillämpningar kan dock datamängden vara svår och/eller dyr att framställa vilket leder till att man tvingas kompromissa mellan tränings- och

testmängd och gör det svårt att göra omfattande hold-out-test. I ProActives fall kräver varje observation att man har identifierat eller uteslutit cancer och dess stadie hos en patient, tagit blodprov och gjort en assay på de 180 biomarkörerna vilket gör det omöjligt att samla ihop hundratusentals observationer inom rimlig tid och budget.

2.2 Prestandan hos en designprocedur

Om man väljer att uppskatta prestandan hos en designprocedur istället för en enskild klassificerare öppnar sig en rad nya möjligheter att bättra utnyttja sin data på. Nu talar vi om att uppskatta priorn $p(q|N_d)$ istället för posteriorn $p(q|N_d, N_t, k_t)$, d.v.s. hur bra klassificerare vi kan förvänta oss med vår datamängd och metod i allmänhet.

$$P(q|N_d, N_t, k_t) = \frac{P(k_t|q, N_d, N_t)P(q|N_d)}{P(k_t|N_d, N_t)}$$

För att göra det måste man träna och testa tillräckligt många klassificerare så att de rättvist representerar alla möjliga klassificerare som skulle kunna uppstå med den metod man använder sig av på den underliggande fördelning ens data kommer ifrån.

Känner man till den underliggande fördelningen är problemet lätt eftersom man då kan träna oändligt många klassificerare och testa dem på oändligt mycket oberoende data. I praktiken gör man aldrig det men det kan vara användbart att testa ens procedur på kända påhittade fördelningar för att undersöka om den beter sig som den ska, t.ex. om den konvergerar mot sanna lösningen när den får mer data, om den har något systematiskt fel etc.

Om man inte känner till den underliggande fördelningen får man nöja sig med den data man har och göra det bästa av situationen. Eftersom vi här tränar många klassificerare för att uppskatta prestandan kan vi emellertid använda oss av våra observationer flera gånger i olika kombinationer för att på ett mer effektivt sätt utnyttja dem. Som det nämnts ovan krävs det dock att ens data i grunden är en bra representant för den underliggande fördelningen för att man över huvud taget ska ha en chans att få bra resultat, annars finns det inga verkliga samband att fånga upp.

2.3 Balansen mellan bias och varians

Med *bias* menas alla former av systematiska fel en metod eller modell gör. Den kan t.ex. uppkomma genom att metoden är felkonstruerad, datamängden behandlas på ett missvisande sätt eller att det hänt något under insamlandet av datamängden t.ex. förorenade prover.

När observationer användes fler än en gång tillkommer en risk för bias eftersom modellen kan överanpassas till just den data man har. Ett uppenbart fall är om man använder sig av återläggningsuppskattning av felet (eng. *resubstitution estimation*) där man tränar och testar en klassificerare med samma data. En sådan uppskattning av felsannolikheten blir naturligtvis mer positiv än verkligheten eftersom klassificeraren är optimerad för exakt de observationer som den sedan testas med.

Man kan undvika denna bias genom att endast använda varje observation en enda gång. Det leder å andra sidan till att uppskattningen får högre varians

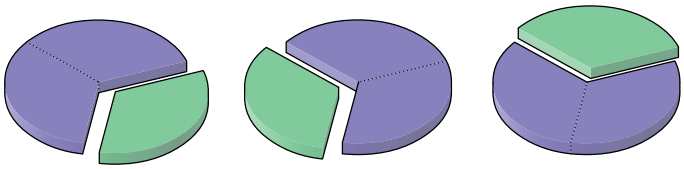
eftersom vi numera har färre observationer att använda oss av vid varje steg i metoden och resultatet därmed blir osäkrare. Det kan dock ofta vara värt att acceptera viss bias för att bekämpa variansen men då ska man vara medveten om att den existerar och helst även dess karaktär (storlek och om den är positiv eller negativ). Gör man det kan man till viss del bekämpa den och tolka resultatet på ett riktigt sätt ändå. Om man t.ex. uppskattat att felsannolikheten hos en designprocedur ligger mellan 10-20% med 99% säkerhet och vet att man har en negativ bias kan man vara säker på att den sanna felsannolikheten i alla fall är lägre än 20%. Positiv bias är svårare att handskas med (att veta att $q > 10\%$ med 99% säkerhet är inte lika användbart) men kan gå att kompensera för. Ett exempel är att om ändå vill återläggninsuppskatta felet kan man göra en bootstrap för att skatta hur optimistiskt testresultatet är [10].

2.4 Populära metoder idag

Idag finns det två grupper av metoder som dominerar när det gäller prestandauppskattning av designprocedurer. Den ena kallas *partitionering* och företräds främst av *korsvalidering* (CV) och den andra för *resampling* och företräds främst av *bootstrap* (BTS).

2.4.1 Korsvalidering

CV är den förmodligen allra vanligaste metoden att uppskatta prestanda med idag [11]. Principen är att dela upp datamängden i ett antal partitioner (disjunkta delmängder av samma storlek) och i tur och ordning använda alla partitioner utom en för träning av en klassificerare och låta den utelämnade partitionen agera testmängd. Om man t.ex. delar datamängden i tre delar kan man träna på del 1 och 2 och testa med del 3, sedan träna en ny klassificerare på del 2 och 3 och testa med del 1 och till sist träna en tredje klassificerare på del 3 och 1 och testa på del 2, se fig. 3.



Figur 3: Uppdelning av datamängd vid trefaldig CV. De blåa segmenten motsvarar träningsmängd och de gröna testmängd.

De tre klassificerarnas sammanlagda felsannolikhet kallas för *korsvalideringsfelet* och ger en uppfattning om hur bra en klassificerare tränad på $2/3$ av datamängden kan förväntas vara. I sin extremaste form används endast en observation i taget för att testa klassificerare tränade på all övrig data, s.k. *leave-one-out-korsvalidering* (LOOCV), vilket i dagens läge har blivit näst intill en standard. Dess popularitet beror både på att metoden är enkel att förstå och att den ger ett svar utan bias för träningsmängder av storleken $N_d = N - 1$ och därav även ger ett svar med liten bias om hela datamängden används till träning ($N - 1 \approx N$) vilket är naturligt att man tränar den slutgiltiga klassificeraren på. CV lider däremot av en hög varians som kan orsaka problem istället.

2.4.2 Resampling

Resampling eller *återsampling* innebär att man drar många tränings- och testmängd från sin datamängd med återläggning och på så sätt återanvänder det upprepade gånger. Det här leder till att man får många fler testresultat som kan användas för att uppskatta prestandan vilket ger en lägre varians men också en ökad optimistisk bias. Hur stor den blir beror på hur mycket tränings- och testmängderna överlappar, jämför CVs partitioner som inte överlappar alls (ingen bias) med återuppläggningsuppskattning som överlappar helt (mycket bias). Biasen kan kontras genom att tränings- och testmängder dras från oberoende delmängder av datamängden, alternativt bara återsampla träningsmängder från en del av datamängden och låta resten utgöra en fast testmängd.

Det finns en uppsjö av förfaranden att uppskatta prestandan hos klassificerare tränade på återsamplad data på. Några vanliga är BTS, jackknife och 632-uppskattning om vilka man kan läsa mer om i Hand (1986) [10]. Känner man till hur datamängden är sammansatt kan man även utnyttja det och göra en skräddarsydd återsamplingsmetod för att bättre passa problemet och öka prestandan. Ett exempel är den form av BTS som används för att testa trovärdigheten hos fylogenetiska träd baserat på sekvensdata [4].

3 Problem med dagens metoder

Det finns två stora problem med dagens prestandauppskattning. Dels redovisar inte CV och BTS i sina standardutföranden någon form av spridningsmått för den uppskattade felsannolikheten och dels råder det stor okunskap bland de som gör vetenskapliga undersökningar om hur klassificerare och prestandautvärderingsmetoder bör användas. Det händer allt för ofta att man bara har ett litet antal observationer, använder en designprocedur som korsvalideras och tränar en klassificerare på all data med korsvalideringsfelet redovisat som den slutliga felsannolikheten. Så får man naturligtvis inte göra men tyvärr förekommer det ibland även på hög vetenskaplig nivå.

Den intuitiva uppfattningen om hur mycket data det behövs för att träna en pålitlig klassificerare är ofta avsevärt lägre än vad som verkligen är fallet. Det kan t.ex. tyckas att 100 bekräftade patientfall och en lika stor kontrollgrupp i en studie av en sjukdom kan verka mycket men det kan vara långt ifrån tillräckligt beroende på den underliggande fördelningen, problemets dimension m.m. Det kommer man däremot inte bli varse om ifall man använder CV eller BTS för att uppskatta prestandan eftersom de återigen inte redovisar någon spridning i svaret. Vår nya metod eller ett stort hold-out-test gör däremot det och är säkrare men mer krävande alternativ.

4 Metod

Proceduren som utvecklats består i grova drag av nedanstående steg, se även fig 4. På sätt och vis kan man se den som en kontinuerlig motsvarighet till BTS med oberoende replikat där alla testfel används och inte bara testfelens medelvärde.

- Datamängden normaliseras och delas därefter i ett antal delar, s.k. *bags*.

- För varje bag byggs en bayesiansk kärnskattning av fördelningen, se stycke 4.2.

- Från kärnskattningen bolsteras (dras) ett stort antal träningsmängder av fast storlek samt en stor testmängd på vilka det tränas och testas klassificerare.

- I varje bag fås då ett testfel per klassificerare vilka används för att bygga ett histogram över klassificerarnas prestanda. Histogrammet är en uppskattning av designprocedurens prestanda på den okända fördelningen, d.v.s. dess prior. Är datamängden tillräckligt stor bör kärnskattningen gå mot den sanna fördelningen och histogrammet mot den sanna priorn.

- Genom att göra ett nytt histogram på alla testfel från alla bagar erhålls en slutgiltig uppskattning av priorn. Ett osäkerhetsmått fås genom att jämföra hur mycket priorn skiljer sig mellan de olika bagarna.

- Följande sista steg räknar ut den sanna priorn och genomförs bara om man känner till den sanna fördelningen. Det kan alltså endast tillämpas på teoretiska problem med syfte att övertyga sig om att metoden fungerar och inte på empirisk data.

  Från den sanna fördelningen dras ett stort antal träningsmängder av samma storlek som de bolsterade träningsmängderna och en stor testmängd på vilka det tränas och testas klassificerare. Av deras testfel bildas ett histogram som representerar den sanna priorn mot vilken de uppskattade kan jämföras.

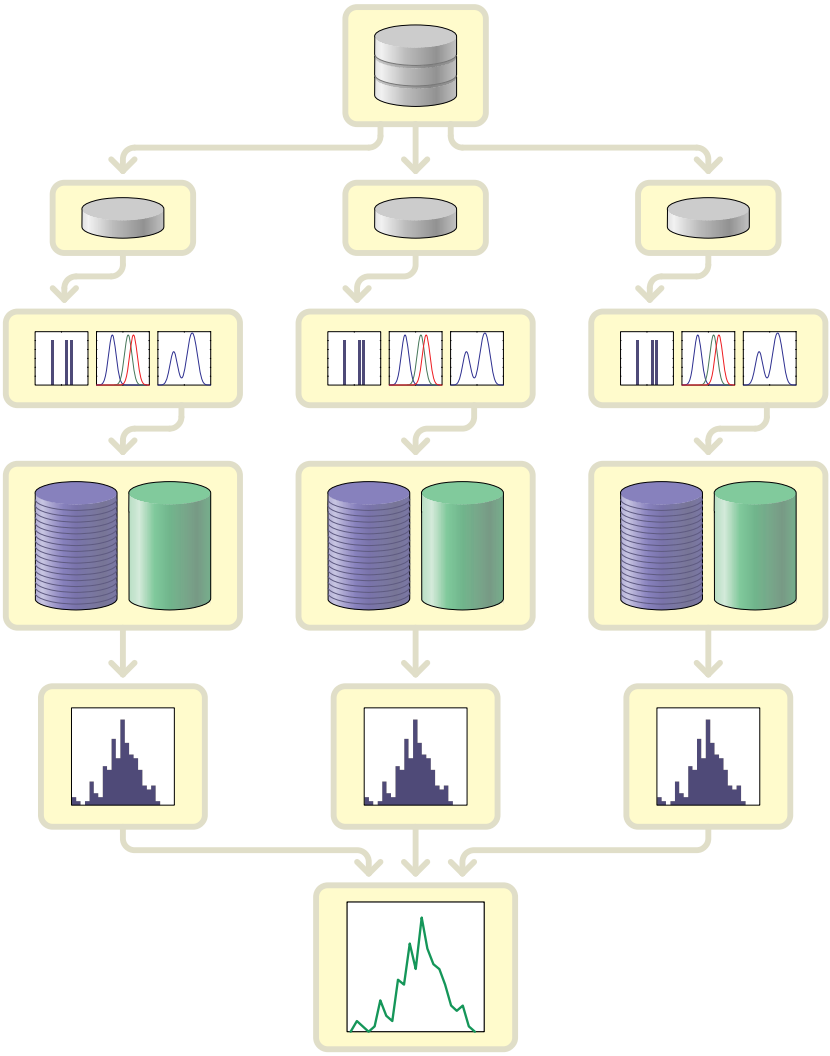
4.1 Kärnskattning och bolstering

En *kärnskattning* $p(\overline{x}|D)$ av en okänd fördelning är en kontinuerlig skattning av fördelningen baserad på ett stickprov D från den. En sådan konstrueras genom att det runt varje observation $\overline{x}_n$ i stickprovet läggs en *kärna*, en funktion som är större än 0 i ett område runt observationen och lika med 0 längre bort. Tillsammans bildar alla kärnor en skattning av hela fördelningen, se ekv. 3 och fig. 5.

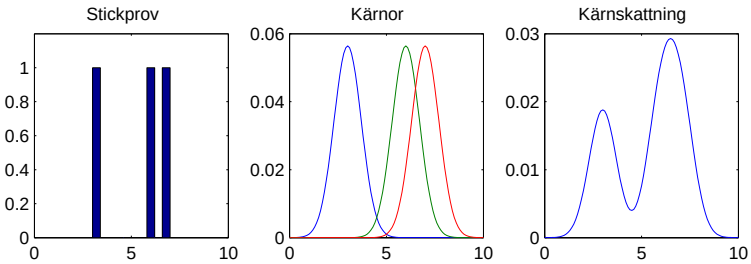
$$p(\overline{x}|D) = \frac{1}{N}\sum_{n=1}^{N} K(\overline{x} - \overline{x}_n) \qquad \begin{cases} K(\overline{x}) > 0 & \|\overline{x}\| \text{ nära } 0 \\ K(\overline{x}) = 0 & \|\overline{x}\| \gg 0 \end{cases} \tag{3}$$

Från kärnskattningen kan sedan en ny observation dras genom att man väljer en av de ursprungliga observationerna med likformig slump och därefter slumpar ett tal från den fördelning dess kärna definierar. Detta kallas för att *bolstera* observationer och kan ses som en kontinuerlig motsvarighet till återsampling. I ProActives fall finns det två stora fördelar med den här tekniken jämfört med traditionell återsampling.

- Modellen beskriver verkligheten bättre än traditionell återsampling som inte fungerar särskilt väl på kontinuerlig data. I ProActives fall består datamängden av biomarkörernas koncentrationer och det vore märkligt att genom BTS anta att de bara kan anta just de värden som observerats i



Figur 4: En grafisk överblick över den utvecklade metoden.



Figur 5: 3 observationer har dragit från en okänd fördelning. Runt observationerna läggs kärnor som sedan slås ihop till en fördelning som skattar den okända fördelningen.

stickprovet. Deras underliggande fördelningar skulle då antas vara 0 överallt utom i de punkter en observation förekommer där den istället skjuter i höjden. Om man istället antar att fördelningen är större än noll i ett område runt punkten får man en mer trovärdig fördelning.

- Från kärnskattningen kan man dra oändligt många nya observationer att träna klassificerare på. Därmed är problemet med tillgång på data för träning och test löst men å andra uppstår ett nytt problem i att göra en bra kärnskattning.

För att göra en kärnskattning behöver man bestämma hur ens kärnor ska se ut. Oftast brukar man göra skillnad på kärnornas funktion K (vilken form kärnan får) och bredd h eftersom formen beror på problemets natur och bredden på hur mycket data man har.

4.1.1 Val av kärnfunktion

Valet av funktion bör göras med avseende på hur man tror den fördelning man vill skatta ser ut. Den överlägset vanligaste kärnfunktionen och som även används i det här arbetet är en *gaussklocka*[2] men man kan använda vilken funktion som helst förutsatt att den har en begränsad area. Ekv. 4–6 och fig. 6 visas ett par exempel på kärnfunktioner och kärnskattningar gjorda med dem utifrån samma stickprov. Anledningen till att gaussklockan är så vanlig som kärnfunktion är flera: den definieras av endast två parametrar per dimension som båda är lätta att skatta, enligt centrala gränsvärdessatsen går en summa av stokastiska variabler mot en normalfördelning med ökande antal termer [2] och de ger mjuka gradienter snarare än skarpa linjer i den skattade fördelningen vilket i allmänhet brukar passa bättre. I ProActives fall är alla tre bra argument för att använda gaussklockan som kärnfunktion.

$$K_{\text{Fyrkantig}}(x) = \begin{cases} \frac{1}{h} & |x| \leq \frac{h}{2} \\ 0 & |x| > \frac{h}{2} \end{cases} \tag{4}$$

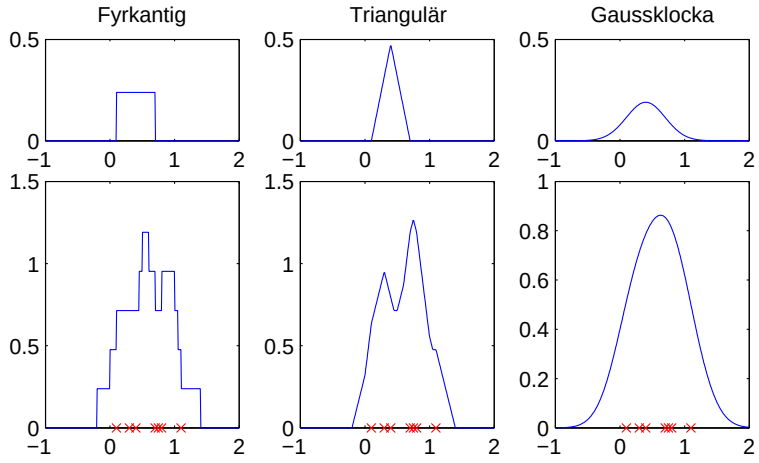
$$K_{\text{Triangulär}}(x) = \begin{cases} \frac{2}{h}\left(1 - \frac{2}{h}|x|\right) & |x| \leq \frac{h}{2} \\ 0 & |x| > \frac{h}{2} \end{cases} \tag{5}$$

$$K_{\text{Gaussklocka}}(x) = \frac{1}{h\sqrt{2\pi}} \exp\left(-\frac{x^2}{2h^2}\right) \tag{6}$$

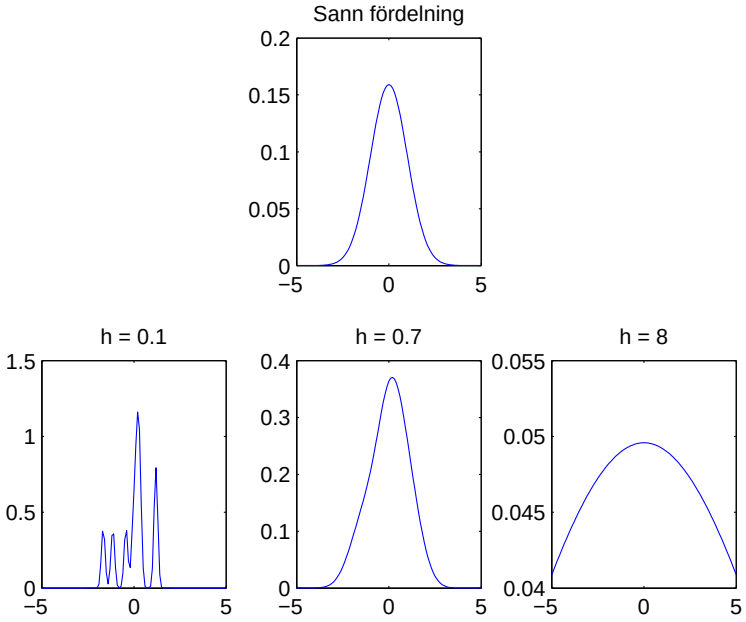
4.1.2 Val av bredd

Att välja en lämplig bredd h (eng. smoothing) för kärnorna är i regel ett svårare problem än att välja funktion. Har man lite data bör bredden vara stor så att inte kärnorna är isolerade från varandra och har man mycket data bör bredden vara liten så att man inte suddar ut fördelningen för mycket. Fig. 7 visar tre fall där bredden är för liten, lagom och för stor. Flera metoder har föreslagits för att hitta ett lämpligt värde på h och Webb redogör för några av dem [16]. Den metod som används i det här arbetet kallas för *bayesiansk kärnskattning* och

[2] Termerna gaussklocka, gaussfunktion, normalkurva och normalfördelning brukar ofta användas på samma saker vilket kan vara lite förvirrande. I det här arbetet används *normalfördelning* för att beskriva fördelningen och *gaussklocka* för att beskriva dess täthetsfunktion.



Figur 6: I den övre raden visas tre olika kärnfuntioner och i den under raden visas deras respektive kärnskattningar på 7 observationer.



Figur 7: Ett stickprov på 10 observationer har dragits från den sanna fördelningen utifrån vilket kärnskattningar har gjorts för 3 olika kärnbredder h. Kärnfunktionen $K(x)$ som använts är gaussklockor där standardavvikelsen σ agerar breddvariabel ($h = \sigma$). Det syns tydligt att valet av bredd har stor inverkan på skattningens kvalité.

finns beskriven i stycke 4.2. Då kärnskattningarna i det här arbetet konstrueras med guassklockor har vi valt att använda deras spridning σ (motsvarande standardavvikelse för en normalfördelning) som breddvariabel $h = \sigma$. Hädan efter kommer endast σ användas eftersom det känns mest naturligt.

4.2 Bayesiansk kärnskattning

Tanken bakom bayesiansk kärnskattning är att man inte vill begränsa sig till en kärnbredd utan istället integrera över alla tänkbara bredder och göra en kombinerad fördelning baserad på hur mycket stöd det finns för varje bredd i datamängden. Kärnskattningen för en given bredd visas i ekv. 7 och för samtliga möjliga bredder i ekv. 8. I praktiken är det dock betydligt lättare att istället räkna ut kärnskattningen som en diskret approximation av ekv. 8 som då ser ut som ekv. 9.

$$p(\overline{x}|\sigma) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{\sigma^d\sqrt{2\pi}}\exp\left(-\frac{\|\overline{x}-\overline{x}_n\|^2}{\sigma^2}\right) \tag{7}$$

$$p(\overline{x}|D) \equiv \int_0^\infty p(\overline{x}|\sigma, D)p(\sigma|D)d\sigma \tag{8}$$

$$p(\overline{x}|D) \approx \Delta\sigma\sum_{m=1}^{M} p(\overline{x}|\sigma_m, D)p(\sigma_m|D)d\sigma \tag{9}$$

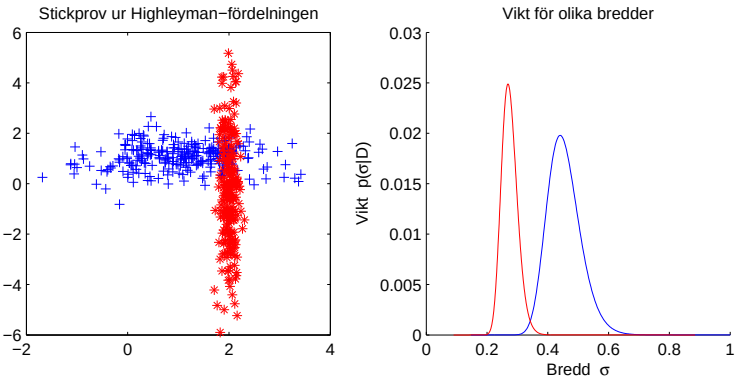
Hur mycket stöd det finns för varje vikt i datamängden $p(\sigma|D)$ kan räknas ut på olika sätt men i grund och botten går det alltid ut på att jämföra hur troligt det är att ett antal datapunkter D_1 dragits från en fördelning som spänns upp av ett antal andra datapunkter D_2 för olika σ. Det som skiljer metoderna åt är framför allt hur D_1 och D_2 väljs men då de baseras på Bayes sats (ekv. 10) har även priorn $p(\sigma|D_2)$ viss betydelse, se stycke 4.2.3. Datamängdens sannolikhet $p(D)$ beror inte av σ så den används som normaliseringskonstant.

$$p(\sigma|D) = p(\sigma|D_1, D_2) = \frac{p(D_1|\sigma, D_2)p(\sigma|D_2)}{p(D)} \tag{10}$$

Nedan redovisas två metoder för att räkna ut $p(\sigma|D)$ som mynnar ut i ekv. 11 och 12. Gemensamt för båda är att desto tätare det är mellan datapunkterna desto lägre standardavvikelser kommer favoriseras vilket känns naturligt. Fig. 8 visar ett exempel på detta där vikter för olika bredder räknats ut för ett stickprov ur Highleyman-fördelningen.

4.2.1 Ordinarie bestämning av $p(\sigma|D)$

Det mest rigorösa sättet att bestämma $p(\sigma|D)$ på är att dela datamängden i två disjunkta delar där den ena spänner upp en fördelning och den andra avgör hur sannolika olika σ är. Då används varje observation endast en gång och man riskerar inte att få någon bias.



Figur 8: Utifrån ett stickprov med 100 observationer (50 per klass) har olika bredders vikter räknats ut på det ordinarie sättet beskrivet i stycke 4.2.1.

$$
\begin{aligned}
p(D_1|\sigma, D_2) &= \prod_{i=1}^{N_{D_1}} p(\overline{x}_i|\sigma, D_2) \\
p(\overline{x}_i|\sigma, D_2) &= \frac{1}{N_{D_2}} \sum_{j=1}^{N_{D_2}} K(\overline{x}_i - \overline{x}_j) \\
p(\sigma|D_1, D_2) &= \prod_{i=1}^{N_{D_1}} \left[\frac{1}{N_{D_2}} \sum_{j=1}^{N_{D_2}} K(\overline{x}_i - \overline{x}_j) \right] \frac{p(\sigma|D_2)}{p(D_1|D_2)} \qquad (11) \\
&= \prod_{i=1}^{N_{D_1}} \left[\frac{1}{N_{D_2}} \sum_{j=1}^{N_{D_2}} \frac{\alpha}{\sigma^d} \exp\left(-\frac{\|\overline{x}_i - \overline{x}_j\|^2}{2\sigma^2} \right) \right] \frac{p(\sigma|D_2)}{p(D_1|D_2)}
\end{aligned}
$$

4.2.2 Leave-One-Out-bestämning av $p(\sigma|D)$

Ett mer effektivt sätt att utnyttja datamängden på men som introducerar en risk för bias är att välja D_1 och D_2 på ett sätt som liknar LOOCV (se stycke 2.4.1). Vikten $p(\sigma|D)$ bestäms av att man i tur och ordning testar hur sannolikt det är att varje observation $\overline{x}_i$ kan dras från en fördelning som spänns upp av de övriga observationerna $(D - \overline{x}_i)$ med aktuellt σ.

Anledningen till att $\overline{x}_i$ själv inte får vara med och uppskatta sin egen fördelning är att vikten då alltid skulle öka i storlek då σ går mot 0. Eftersom $\overline{x}_i$ då alltid skulle hamna mitt i en av fördelningens toppar skulle man favorisera en fördelning som skjuter i höjden i varje datapunkt och är 0 överallt annars. Den fördelningen är i själva verket samma som BTS-fördelningen så att ta med $\overline{x}_i$ själv skulle leda till att hela metoden blev en BTS med replikat. Det är osannolikt att en verklig fördelning skulle se ut på det viset vilket var själva anledningen till att bolstering används istället för BTS i första taget.

Formeln för LOO-bestämning av vikterna blir därför som ekv. 12 nedan.

$$
\begin{aligned}
p(D|\sigma) &= \prod_{i=1}^{N_D} p(\overline{x}_i|\sigma, D-\overline{x}_i) \\
p(\overline{x}_i|\sigma, D-\overline{x}_i) &= \frac{1}{N_D-1}\sum_{\substack{j=1\\ j\neq i}}^{N_D} K(\overline{x}_i-\overline{x}_j) \\
p(\sigma|D) &= \prod_{i=1}^{N_D}\left[\frac{1}{N_D-1}\sum_{\substack{j=1\\ j\neq i}}^{N_D} K(\overline{x}_i-\overline{x}_j)\right]\frac{p(\sigma)}{p(D)} \qquad (12)\\
&= \prod_{i=1}^{N_D}\left[\frac{1}{N_D-1}\sum_{\substack{j=1\\ j\neq i}}^{N_D} \frac{\alpha}{\sigma^d}\exp\left(-\frac{\|\overline{x}_i-\overline{x}_j\|^2}{2\sigma^2}\right)\right]\frac{p(\sigma)}{p(D)}
\end{aligned}
$$

4.2.3 Val av prior

Eftersom σ är en skalparameter har dess prior valts till $p(\sigma) = \sigma^{-1}$, den s.k. *Jeffreys prior*, vilket man normalt sett gör för sådana [12, 13]. Detta kommer ur skalinvariansprincipen som går ut på att problemet inte ska ändras om det skalas om.

4.3 Kärnskattningar med fler frihetsgrader

Hittills har de flesta exempel visats på endimensionell data där det endast finns en parameter som kan varieras men principerna fungerar likadant om det finns flera. Har problemet högre dimension kan man välja olika kärnfunktioner, bredder och vikter i varje dimension och det är också möjligt att ta hänsyn till beroenden mellan dimensionerna. Ett exempel på det finns i Li et al. [14] som gjort en metod som kan anpassa bredden i varje dimension för individuella kärnor och även vikta hela kärnan olika mycket baserat på osäkerheten i dess mätvärden. Att uppskatta individuella vikter för varje dimension kan dock bli mödosamt eftersom man måste integrera över bredden i varje enskild dimension. Jämför ekv. 8 med ekv. 13 här nedan.

$$
p(\overline{x}|D) = \int_0^\infty \dots \int_0^\infty p(\overline{x}|D, \sigma_1, \dots, \sigma_n)p(\sigma_1|D)...p(\sigma_n|D)d\sigma_1 \dots d\sigma_n \qquad (13)
$$

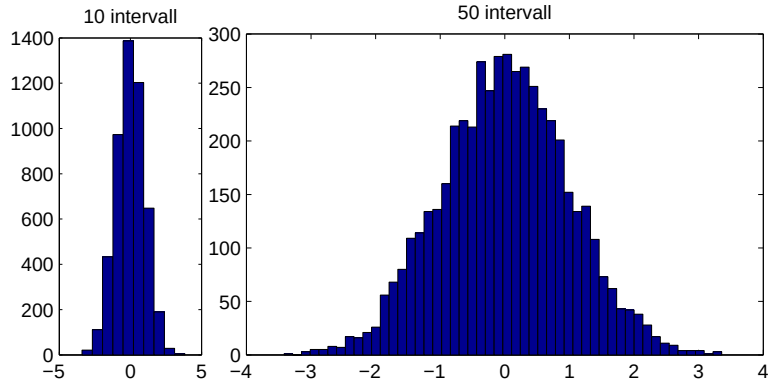
I det här arbetet har det inte tagits någon individuell hänsyn till de olika dimensionerna utan istället hoppas vi att samma vikter ska passa i alla dimensioner efter att datan normaliserats (skalat om till medel 0 och standardavvikelse 1 i samtliga dimensioner). Att använda sig av två olika bredder eller interaktioner mellan dimensionerna i våra testkörningar på 2D-data skulle antagligen inte leda till en tillräcklig ökning i prestanda för att motivera den ökade komplexiteten. Att sedan skala upp och integrera över bredden i alla 180 dimensioner som finns i ProActives datamängd skulle ge en tidskomplexitet på $O(n^{360})$ vilket är fullständigt omöjligt att beräkna.

4.4 Uträkning av skattad och sann prior

För en given typ av klassificerare på en fördelning, sann eller kärnskattad, kan man skatta priorn genom att träna ett stort antal klassificerare på oberoende träningsmängder av samma storlek och testa på en stor oberoende testmängd.

I simuleringarna har alltid testmängder på 10^5 observationer använts vilket garanterar att de uppskattade felsannolikheterna för individuella klassificerare maximalt avviker från sanningen med $\pm 0.41\%$ i maximalt 99% av fallen.

Vid uträkningen av en sann prior har 10^5 klassificerare tränats men i varje bag har bara 5000 klassificerare tränats för att korta ner beräkningstiden. På dessa har histogram med 100 intervall bildats över vilka felsannolikhetsfördelningen allt som oftast koncentrerar sig till 10–50. Detta ger histogram som ser ut som de i fig. 9 och även om de ser lite hackiga ut ger de i alla fall en klar uppfattning om hur priorn ser ut. Ett alternativ för att få jämnare och antagligen bättre priors skulle vara att göra kärnskattningar på testresultaten.



Figur 9: Bilderna visar histogram över stickprov på 5000 observationer från en normalfördelning.

4.5 Implementation

Metoden implementerades i Matlab™[3] med toolboxen PRTools[4] som innehåller funktioner för mönsterigenkänning och klassificeringsproblem [3]. Ett annat (och dessutom gratis) alternativ skulle kunna vara statistikprogrammet R men då alla inblandade hade större erfarenhet av Matlab™ med PRTools och det dessutom redan fanns tillgång på licenser valdes det utan att R undersökts.

Implementationen gjordes i form av körbara m-filer samt en mex-fil skriven i C för att beräkna vikter för olika val av σ. Anledningen till att viktberäkningen lades ut på C-kod är att den kräver att man räknar ut ett stort antal parvisa avstånd mellan observationer vilket leder till hög minnesanvändning om det görs med vanliga matrisberäkningar[5]. Viktberäkningarna fick även logaritmeras

[3] Matlab™ version 7.3.0 (R2006b).

[4] PRTools version 4.0, `http://prtools.org`.

[5] mn för ordinarie uppskattning där m och n är antalet observationer i de olika delmängderna och $\frac{n^2-n}{2}$ för LOO-uppskattning av σ där n är totala antalet observationer.

för att inte hamna utanför datorns beräkningsnoggrannhet, ekv. 14. De kunde sedan skalas upp och normaliseras med algoritmen som följer efteråt.

$$\begin{aligned} p(\sigma|D) &= \frac{p(D|\sigma)p(\sigma)}{p(D)} \\ \ldots &= \prod_{i=1}^{N_D}\left[\frac{1}{N_D-1}\sum_{\substack{j=1\\ j\neq i}}^{N_D} K(\overline{x}_i-\overline{x}_j)\right]\frac{p(\sigma)}{p(D)} \\ \ldots &= \alpha\prod_i\sum_j K(\overline{x}_i-\overline{x}_j)\frac{1}{\sigma}\ , \quad \alpha=\frac{N_D}{(N_D-1)p(D)} \\ \log(p(\sigma|D)) &= \log(\alpha)+\sum_i \log\left(\sum_j K(\overline{x}_i-\overline{x}_j)\right)-\sigma \end{aligned} \tag{14}$$

1. Beräkna $\log(p(\sigma_m|D))$ för alla m med ekv. 14
2. Välj konstanten $c = \max(\log(p(\sigma_m|D))\ , m = 1, 2, ..., M$
3. Definiera ny logaritm $s_m = \log(p(\sigma_m|D)) - c\ , m = 1, 2, ..., M$
4. Beräkna

$$\begin{aligned} p(\sigma_m|D) &= \frac{10^{s_m}}{\sum_{n=1}^{N}10^{s_n}} = \frac{10^{\log(p(\sigma_m|D))-c}}{\sum_{n=1}^{N}10^{\log(p(\sigma_n|D))-c}} = \frac{p(\sigma_m|D)10^{-c}}{\sum_{n=1}^{N}p(\sigma_n|D)10^{-c}} = \ldots \\ \ldots &= \frac{p(\sigma_m|D)}{\sum_{n=1}^{N}p(\sigma_n|D)} \end{aligned}$$

Koden anpassades även för att kunna köras parallellt på flera datorer i kluster vilket reducerar beräkningstiden avsevärt.

5 Resultat

Arbetets resultat kan delas in i en teoretisk del bestående av metoden och dess formler, implementationen samt simuleringsresultaten. Det teoretiska resultaten finns redovisade i del 4 av rapporten och detaljer gällande implementationen kan fås på begäran. Resterande innehåll i resultat-delen handlar om simuleringsresultaten.

5.1 Hur simuleringar och diagram ska tolkas

Med en *körning* avses ett utförande av metoden och för att undvika långa och svårlästa meningar används en standardnotation för att beskriva dess inställningar. När det står att en körning gjorts med *X på Y N x NN* betyder det att klassificeraren som använts är X, den sanna fördelningen är Y och datamängden består av N bags med NN observationer i varje, t.ex. LDC på Lithuanian 8 x

100. Storleken på träningsmängderna N_d varierar men hålls alltid konstant inom samma körning. Övriga parametrar är genomgående samma eller så står det utryckligen att de varierats för att studera dess effekt. Följande standardvärden har använts:

Viktuppskattningsmetod	LOO, se stycke 4.2.2
Breddprior	Jeffreys, se stycke 4.2.3
Antal klassificerare per bag	5000
Storlek på testmängd N_t	10^5 observationer

För att få en uppfattning om hur lika en sann och uppskattad prior är med avseende på form och position har deras histogram jämförts och som ett mer jämförbart mått har avståndet från den sanna priorns 10- och 90-percentiler till varje bags 10- och 90-percentiler räknats ut. Se t.ex. övre grafen i fig. 10 för ett exempel på histogramjämförelse och fig. 12 för percentiljämförelse. Medelavståndet markeras med röd linje i låddiagrammen och innefattar inte outliers som markeras med röda plustecken.

I alla diagram där en sann prior förekommer har den ritats upp med grön färg medan uppskattade priors betecknas med blå färg. Kortfattad info om de klassificerare som förekommer i simuleringarna finns i bilaga A.

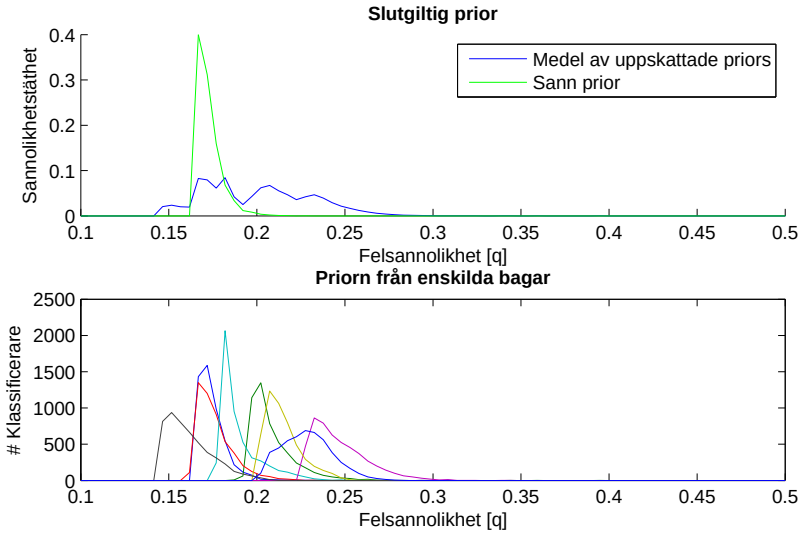
5.2 Övergripande simuleringsresultat

Samtliga simuleringar har gjorts på data från kända fördelningar i antingen en eller två dimensioner. Koden skalar upp väl till de 180 dimensioner som ProActives datamängd kommer ha men då önskad prestanda inte uppnåddes inom ramen för examensarbetet fördrog vi att arbeta i mindre skala.

Simuleringsresultaten visar att koden fungerar som den ska men att de bayesianska kärnskattningarna inte blir tillräckligt lika den sanna fördelningen för att man ska nå fram till den sanna priorn. Fig. 10 visar en typisk körning där priorn i varje bag mer eller mindre liknar den sanna priorn formmässigt men har olika bias vilket får den slutgiltiga priorn att inte likna den sanna. Biasen var nästan alltid negativ men i vissa specialfall hände det att det var positiv. Hade den alltid varit negativt och det kunde visas matematiskt hade metoden kunnat användas för att räkna ut en undre gräns för prestandan men det går alltså inte nu.

Metoden konvergerar visserligen med ökad mängd data men det går långsamt och det verkar inte gå att få bort biasen helt. I stycke 5.3 visas det att biasen troligtvis orsakas av skattningen av fördelningen. Det verkar dock inte spela någon större roll om kärnbreddernas vikter skattats med den ordinarie metoden med delad datamängd eller LOO vilket är en glad nyhet eftersom LOO då ger likvärdig prestanda som den ordinarie metoden fast med bara hälften så mycket data[6]. Körningar med ökande bagstorlek gjordes för fyra olika problem som visas i fig. 11 med två replikat var. De olika problemen konvergerade ungefär lika fort men hade olika bias, huvudsakligen negativ men QDC fick svagt positiv. Resultaten skiljde bara lite mellan replikaten i samtliga fall vilket tyder på att metoden är robust (men det krävs ytterligare försök för att vara säker på det).

[6]Om LOO har tilgång till n observationer kan n-1 av dem spänna upp fördelningen och den sista användas för att räkna ut vikterna medan den ordinarie metoden bara kan låta n/2 spänna upp fördelningen och de andra n/2 räkna ut vikterna.



Figur 10: Den övre bilden visar den slutgiltiga uppskattningen som är ett medelvärde av alla bagars prior (nedre bilden). Den här körningen har gjorts med LDC på Lithuanian 8 x 100.

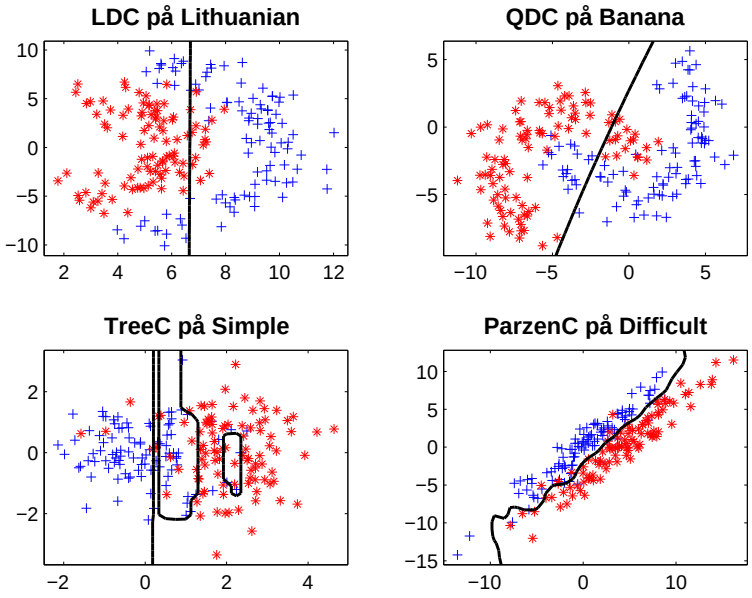
För att ge en uppfattning om hur resultatet ser ut visar fig. 12 och 13 hur det gick i fallet TreeC på Simple.

5.3 Olika världsbilder

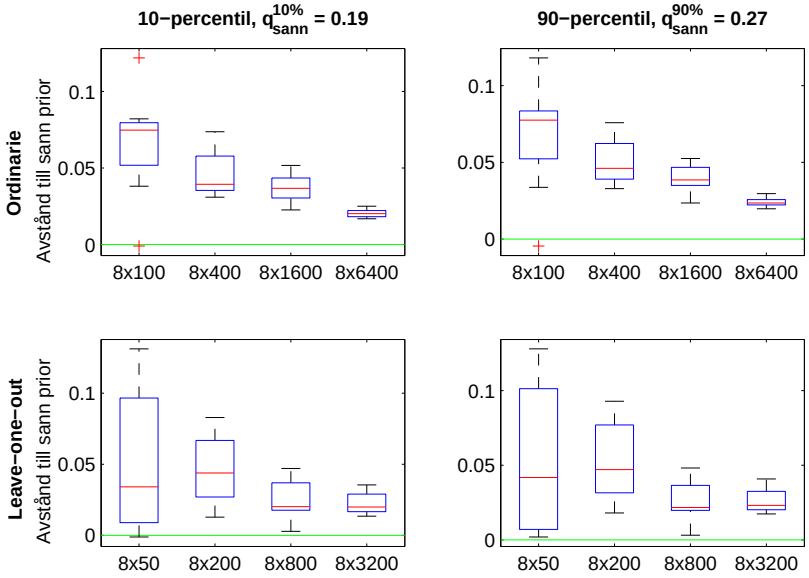
För att bekräfta att implementeringens ramverk inte orsakade någon bias genomfördes ett antal körningar utan kärnskattning och bolstering där data till klassificerarna istället drogs direkt från de sanna fördelningarna. Det ska leda till att den sanna priorn och priorn i varje bag uppskattas på exakt samma sätt fast med olika antal klassificerare. Resultatet skiljde sig en aning mellan bagarna vilket gjorde att de slutgiltiga "uppskattade" priorna är lite slätare än de sanna (se fig. 14) men det ser ändå ut som att ramverket fungerar som det ska. Skillnaden kan bero på att det kanske inte är tillräckligt att träna 5000 klassificerare i varje bag. Percentilavstånden var av storleksordningen 10^{-5}.

Problemet gjordes lite svårare genom att förse metoden med en datamängd dragen från Highleyman-fördelningen, avslöja att de rörde sig om normalfördelad data men låta metoden själv uppskatta fördelningens parametrar för varje klass. Två körningar gjordes med 8x200 och 8x800 observationer för LDC, QDC och ParzenC. Då blev resultatet lite spretiga men percentilernas medelavstånd låg fortfarande på en fullt tillräcklig låg nivå, storleksordning 10^{-3}.

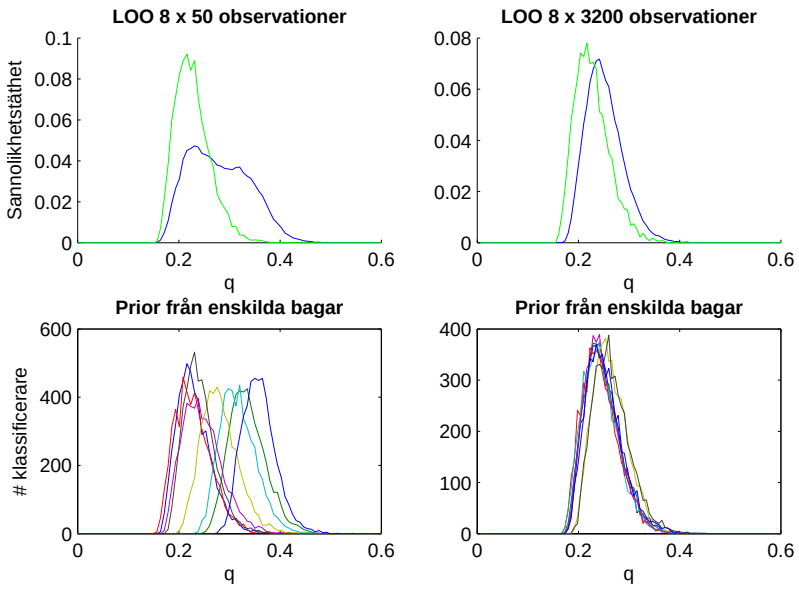
Därefter försvårades problemet ytterligare genom att återgå till att köra metoden i sitt huvudutförande fast på väldigt lätta klassificeringsproblem där klasserna dragits isär, se fig. 15. Det borde ge besultsgränsen möjlighet att variera mer utan att resultatet påverkas så mycket även om fördelningen suddas ut av kärnskattningen. Problemen kördes med kärnskattningar baserade på 40–100 observationer och resulatet för 100 observationer visas i tabell 1. Det gick markant sämre än för körningarna som skattade normalfördelningsparametrar



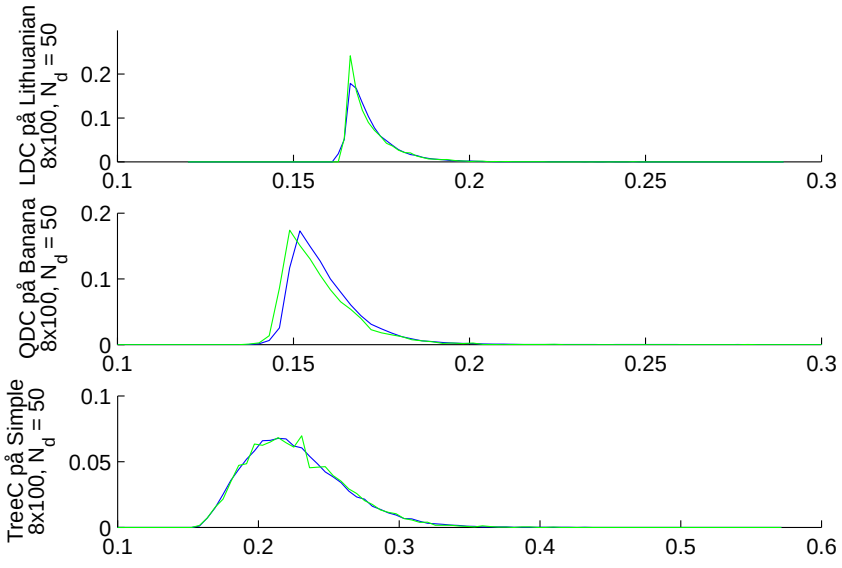
Figur 11: 4 svåra klassificeringsproblem där opassande klassificerare valts med flit. De tjocka svarta linjerna visar vilken typ av beslutsgränser de olika klassificerarna drar.



Figur 12: TreeC på Simple med ökande mängd data. Låddiagrammet visar hur avståndet mellan den uppskattade och sanna priorns 10- och 90-percentil varierar mellan olika bags.

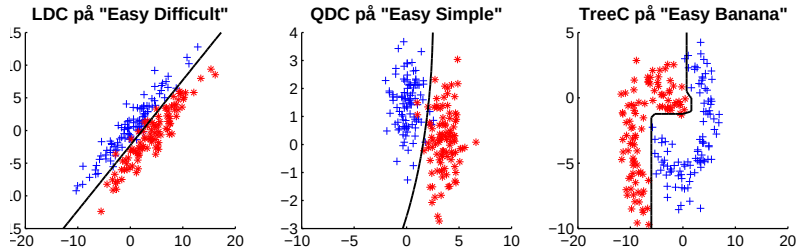


Figur 13: Plot av priorn från LOO-körningarna med minst och mest data i fig. 12.



Figur 14: Tre körningar med data som alltid är dragen från den sanna fördelningen. Anledningen att den sanna priorn i tredje körningen är hackigare än den uppskattade är att den uppskattade är medel av 8 stycken som alla påminde om den sanna i formen.

och liknade de inledande körningarna i stycke 5.2 med långsam konvergens och negativ bias i de flesta fall. Klasserna drogs då isär ytterligare men resultatet förblev det samma. I synnerhet hade LDC svårt att klara av de förenklade Difficult-fördelningarna.



Figur 15: Klassificerare av ovan angivna typer på respektive fördelningar har en medelfelsannolikhet på 3% när de tränas med 100 observationer och en spridning på c:a 0.2% för LDC och QDC och 1.3% för TreeC. De är med andra ord betydligt lättare klassificeringsproblem än de i fig. 11.

	Sann prior		Skattad prior	
	Medel	Std.avvikelse	Medel	Std.avvikelse
LDC	3.4%	0.5%	13.2%	4.9%
QDC	3.4%	0.7%	6.2%	1.6%
TreeC	6.2%	4.3%	7.7%	4.0%

Tabell 1: Resultat från körningarna på problemen i fig. 15.

5.4 Val av breddprior

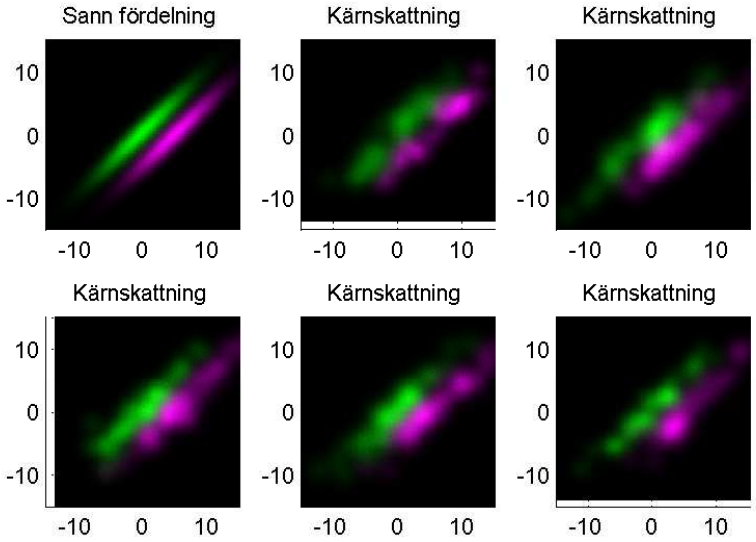
Att prior-uppskattningen för de lätta problemen gick dåligt misstänktes kunna bero på att kärnskattningarna var för suddiga jämfört med de sanna fördelningarna. Därför kördes QDC på "mycket förenklad Simple" om fast med en mycket starkare breddprior $p(\sigma) = \sigma^{-4}$. Detta borde få kärnorna att dra ihop sig och lämna ett större avstånd mellan klasserna vilket borde göra klassificeringsproblemet på kärnskattningarna lättare. Resultatet blev dock ännu sämre och bagarnas medelbias ökade från 1–2% till 2–3% för kärnskattningar baserade på 40–100 observationer. Det gjordes inga vidare undersökningar av breddpriorns inverkan utan istället användes Jeffreys prior till alla simuleringar.

6 Diskussion och slutsatser

6.1 Konvergens och bias

Metoden lider av två problem: långsam konvergens och ojämn bias. Båda dessa beror på att kärnskattningarna inte konvergerar mot den sanna fördelningen tillräckligt fort utan blir suddigare och varierar mellan stickprov om de inte förses med mycket data, se fig. 16 för några exempel. Variationen leder till att klassificeringsproblemen i olika bags skiljer sig från varandra och därför inte

heller har exakt samma prior vilket får den slutgiltiga priorn att konvergera långsamt. Suddigheten försvårar problemen vilket leder till att klassificerarna oftast presterar sämre än de hade gjort på den sanna fördelningen, därav den negativa biasen.



Figur 16: Den första bilden visar den förenklade Difficult-fördelningen och de andra 5 visar bayesianska kärnskattningar baserade på 100 observationer från den.

Hur mycket uppskattningen lider av detta beror på hur känsligt originalproblemet är för utsuddning och slumpmässig variation. Fallet LDC på förenklad Difficult är mycket känsligt eftersom det bara är liten skillnad mellan klasserna och LDC är perfekt lämpad för att hitta den. QDC på Banana å andra sidan (se fig. 11) har ingen uppenbar bias alls eftersom det är omöjligt för QDC att hitta den optimala beslutsgränen och därför inte lider lika mycket av att problemet suddas ut. Däremot kvarstår den långsamma konvergensen eftersom kärnskattningarna fortfarande har svårt att hitta fram till den sanna fördelningen med lite data och problemet får varierande svårighetsgrad i de olika bagarna.

För att göra en bra kärnskattning måste man utgå från ett representativt stickprov från den sanna fördelningen vilket har visat sig kräva mer data än förväntat. En någorlunda komplicerad fördelning med flera toppar eller snabba förändringar i täthet kräver hundratals observationer redan i en dimension och tusentals i två. Vi hade däremot hoppats att enklare och antagligen mer realistiska fördelningar skulle vara lättare att uppskatta och att de i alla fall skulle ge någorlunda likvärdiga klassificeringsproblem som de sanna fördelningarna men det visade sig variera mycket från fall till fall.

6.2 Anpassningar efter lokal täthet

En aspekt som den bayesianska kärnskattningsmetoden inte tar hänsyn till är fördelningens varierande täthet utan alla kärnor får samma utseende oavsett om de befinner sig i ett område med många eller få observationer. Bredden blir då

en kompromiss mellan fördelningens täta och glesa områden vilket leder till att de täta blir utplattade och suddigare än de borde och de glesa blir grynigare än de borde. De täta områdena lider antagligen inte nämnvärt av utsuddningen eftersom de ändå prioriteras högre än de glesa p.g.a. de innehåller fler observationer som påverkar vikterna. Beslutsgränsen bör även gå långt därifrån för annars har man antagligen valt olämpliga särdrag, i ProActives fall markörprotein som inte är kopplade till sjukdomen. Däremot kan gryniga glesa områden vilseleda klassificerarna och leda till att skattningarnas utseende varierar onödigt mycket mellan bagarna. En möjlig lösning på det här problemet kan vara att använda en *adaptiv kärnskattningsmetod* som kan anpassa sin bredd lokalt efter hur tätt det är mellan observationerna. Det är ett område som studerats mycket bl.a. av Abramson (1982)[1], Hall och Marron (1988)[9] och Sain (1994)[15].

6.3 Utdragna fördelningar

En typ av specialfördelningar som vår metod har svårt att klara av är sådana som är smala och utdragna i flera dimensioner, som Highleyman-fördelningen eller Difficult-fördelningen. På dem får våra cirkulära kärnor svårt att anpassa sin bredd och resultatet blir ofta både lite suddigt och grynigt på samma gång. Anledningen till att vi ändå valt att använda oss av cirkulära kärnor är för att vi tror att sådana fördelningar är relativt osannolika att stöta på i verkligheten och vi föredrog att hålla metoden enkel. Det skulle dock vara skönt att veta att de inte vållar några problem ifall de trots allt dyker upp. Highleyman-typen skulle kunna klaras av genom att ha en metod som stöder olika bredder i olika dimensioner, kanske även adaptivt, utan att bli allt för beräkningsintensiv. Kovariansproblemet går att lösa genom principalkomponentanalys (PCA) [11] innan datan normaliseras. Nackdelen med PCA är att då även den slutgiltiga klassificeraren kan behöva tränas på PCA-transformerad data vilket gör den svårtolkad. Om man t.ex. använder sig av beslutsträd måste man göra det eftersom den är starkt beroende av vilka särdrag man använder medan diskriminanter (LDC och QDC) inte är det.

7 Framtida utveckling

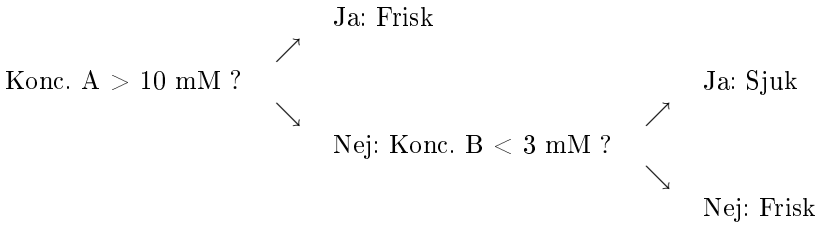
När väl ProActives riktiga data kommer ska det bli intressant att se om de påhittade teoretiska problemen i det här arbetet är realistiska och därefter anpassa metoden efter hur verkligheten ser ut. Under tiden skulle man kunna leta efter liknande äldre data men utbudet av sådan är väldigt begränsad. Fokus kommer därför först läggas på att förbättra kärnskattningsmetoden och bygga in stöd för täthetsanpassning eftersom det är en aspekt som inte studerats ordentligt än. Att bygga in stöd för PCA kan också bli aktuellt samt att konstruera priors med kärnskattningar istället för histogram.

A Klassificerare

- LDC, linjär diskriminant. Beslutsgränsen baseras på en viktad summa av särdrag som bäst separerar klasserna och ser i 2D ut som en rak linje.

- QDC, kvadratisk diskriminant. Beslutsgränsen räknas ut på liknande sätt som för LDC men summan består både av kvadratiska och linjära termer. Gränsen ser i 2D är ett kägelsnitt vilket kan anta formen av en rak linje, parabel, hyperbel, cirkel eller ellips.

- TreeC, beslutsträd. Delar datarymden upprepade gånger med regler vilket ger raka beslutsgränser som går vinkelrätt mot axlarna.



- ParzenC, Parzens klassificerare. Gör en egen kärnskattning utifrån träningsdatan och beräknar den Bayes-optimala beslutsgränsen. En mycket flexibel klassificerare.

Fig. 11 på sidan 21 visar ett exempel på varje klassificerares beslutsgränser.

Referenser

[1] Ian S. Abramson. On bandwidth variation in kernel estimates – a square root law. *The Annals of Statistics*, 10(4):1217–23, 1982.

[2] Gunnar Blom. *Sannolikhetsteori med tillämpningar*. Studentlitteratur, second edition, 1984.

[3] Robert P.W. Duin, P. Juszczak, D. de Ridder, P. Paclik, E. Pekalska, and D.M.J. Tax. Prtools4, a matlab toolbox for pattern recognition. Technical report, Delft University of Technology, Delft, the Netherlands, 2004.

[4] Joseph Felsenstein. Confidence limits on phylogenies: An approach using the bootstrap. *Evolution*, 39(4):783–91, 1985.

[5] Simon Fredriksson, William Dixon, Hanlee Ji, Albert C Koong, Michael Mindrinos, and Ronald W Davis. Multiplexed protein detection by proximity ligation for cancer biomarker validation. *Nature Methods*, 4(4):327–9, 2007.

[6] Simon Fredriksson, Mats Gullberg, Jonas Jarvius, Charlotta Olsson, Kristian Pietras, Sigrún Margrét Gústafsdóttir, Arne Östman, and Ulf Landegren. Protein detection using proximity-dependent dna ligation assays. *Nature Biotechnology*, 20(5):473–7, 2002.

[7] Simon Fredriksson, Joe Horecka, Odd Terje Brustugun, Joerg Schlingemann, Albert C. Koong, Rob Tibshirani, , and Ronald W. Davis. Multiplexed proximity ligation assays to profile putative plasma biomarkers relevant to pancreatic and ovarian cancer. *Clinical Chemistry*, 54(3):582–9, 2008.

[8] Mats Gullberg, Sigrún Margrét Gústafsdóttir, Edith Schallmeiner, Jonas Jarvius, Mattias Bjarnegård, Christer Betsholtz, Ulf Landegren, and Simon Fredriksson. Cytokine detection by antibody-based proximity ligation. *PNAS*, 101(22):8420–4, 2004.

[9] Peter Hall and J.S. Marron. Variable window width kernel estimates of probability densities. *Probability Theory and Related Fields*, 80(1):37–49, 1988.

[10] David J. Hand. Recent advances in error rate estimation. *Pattern recognition letters*, 4(5):335–46, 1986.

[11] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer series in statistics. Springer, fourth edition, 2001.

[12] Edwin T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[13] Harold Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 186(1007):453–61, 1946.

[14] Yunlei Li, Dick de Ridder, Robert P.W. Duin, and Marcel J.T. Reinders. Integration of prior knowledge of measurement noise in kernel density classification. *Pattern Recognition*, 41(1):320–30, 2007.

[15] Stephan R. Sain. *Adaptive Kernel Density Estimation*. PhD thesis, Rice University, Houston, Texas, USA, 1994.

[16] Andrew R. Webb. *Statistical Pattern Recognition*. John Wiley and Sons Ltd, second edition, 2002.