ARAVQ som datareducerare för en klassificeringsuppgift inom datautvinning

HS-IKI-MD-04-106

Niclas Ahlén (a00nicah@student.his.se)

Institutionen för kommunikation och information
Högskolan i Skövde, Box 408
S-54128 Skövde, SWEDEN

Examensarbete på det datavetenskapliga programmet under vårterminen 2004.

Handledare: Kim Laurio

ARAVQ som datareducerare för en klassificeringsuppgift inom datautvinning

Examensrapport inlämnad av Niclas Ahlén till Högskolan i Skövde, för Magisterexamen (M.Sc.) vid Institutionen för Datavetenskap.

2004-09-05

Härmed intygas att allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och att inget material är inkluderat som tidigare använts för erhållande av annan examen.

Signerat: _____

ARAVQ som datareducerare för en klassificeringsuppgift inom datautvinning

Niclas Ahlén (a00nicah@student.his.se)

Sammanfattning

Adaptive Resource Allocating Vector Quantizer (ARAVQ) är en teknik för datareducering för mobila robotar. Tekniken har visats framgångsrik i enkla miljöer och det har spekulerats i att den kan fungera som ett generellt datautvinningsverktyg för tidsserier. I rapporten presenteras experiment där ARAVQ används som datareducerare på en artificiell respektive en fysiologisk datamängd inom en datautvinningskontext. Dessa datamängder skiljer sig från tidigare robotikmiljöer i och med att de beskriver objekt med diffusa eller överlappande gränser i indatarymden. Varje datamängd klassificeras efter datareduceringen med hjälp av artificiella neuronnät. Resultatet från experimenten tyder på att klassificering med ARAVQ som datareducerare uppnår ett betydligt lägre resultat än om ARAVQ inte används som datareducerare. Detta antas delvis bero på den låga generaliserbarheten hos de lösningar som skapas av ARAVQ. I diskussionen föreslås att ARAVQ skall kompletteras med en funktion för grannskap, motsvarande den som finns i Self-Organizing Map. Med ett grannskap behålls relationerna mellan de kluster som ARAVQ skapar, vilket antas minska följderna av att en beskrivning hamnar i ett grannkluster.

Nyckelord: Datautvinnig, Datareduktion, Klassificering, Robotik, Adaptive Resource Allocating Vector Quantizer, Klustring, Tidsserier

Innehållsförteckning

1 Introduktion

Datautvinning är en process för att plocka fram mönster ur stora mängder data, vilka kan tolkas och omvandlas till kunskap om det som datamängderna beskriver. Denna kunskap kan sedan generaliseras och vara underlag för beskrivning och förutsägelse av ny data. Många företag har upptäckt datautvinningens möjligheter till att förutsäga trender, kundbeteenden et cetera och beslutat sig för att investera i tekniken (Pyle, 1999; Han & Kamber, 2001). Företagen har ofta databaser med information som samlats in med vetskapen om att informationen är värdefull, men utan att veta exakt hur den skall utnyttjas. Datamängdernas omfattning kan variera i såväl antal tupler som tuplernas storlek, från några enstaka kilobyte och attribut till flera terabyte och hundratals attribut. Ofta är datamängderna inte anpassade för datautvinning och kan innehålla brus, saknade värden eller vara inkonsekventa. Tillsammans leder dessa faktorer till att mönstren göms i datamängden och försämrar förutsättningarna för en datautvinningsalgoritm.

Genom att förbereda datamängderna görs informationen som finns gömd inom dem mer tillgänglig för datautvinningsverktyget. Detta underlättar arbetet för datautvinningsalgoritmen och ökar chanserna att hitta relevanta mönster. Reinartz (2002) nämner datareduktion som en speciellt lovande ansats för att förbereda datamängder, men pekar också på att dessa tekniker inte är tillräckligt utvecklade inom datautvinning och menar att mer forskning behövs inom området. Han nämner speciellt maskininlärning som en möjlig inspirationskälla.

Linåker (2003) presenterar en teknik för datareducering inom robotikdomänen. Tekniken består av tre steg, där det första steget innebär att indatamängden klustras med hjälp av Adaptive Resource Allocating Vector Quantizer (ARAVQ). ARAVQ klustrar indatamängden och ersätter varje flerdimensionellt exempel med en betydligt enklare beteckning. Den skiljer sig från andra motsvarande tekniker bland annat genom att antalet kluster inte bestäms i förväg utan beror på datamängdens komplexitet, samt genom att ha en kort inlärningstid. Linåker (2003) pekar på teknikens framgångar inom robotikdomänen och menar att den har potential som ett generellt datautvinningsverktyg för tidsserier.

Tidsserier är speciellt svårarbetad data inom datautvinning eftersom de har en dimension – tiden – som begränsar möjligheterna till förberedelse av datamängden (Pyle, 1999). Det borde därför vara eftertraktat med datareduceringsalgoritmer som fungerar väl på just denna typ av data inom datautvinning. Förutsättningarna inom ett typiskt datautvinningsproblem kan dock skilja sig från de förutsättningar som Linåker givit ARAVQ. Exempelvis har de miljöer som ARAVQ testas på i Linåker (2003) en betydligt enklare struktur än de datamängder som används inom datautvinning. Vidare testas det inte hur väl lösningarna från ARAVQ kan generaliseras till nya situationer. Just generaliserbarheten är en mycket viktig egenskap för lösningar inom datautvinning i och med att det oftast är ny, tidigare osedd, data man vill uttala sig om.

Denna studies mål är att testa ARAVQ i en datautvinningskontext genom att undersöka hur den hanterar datamängder med högre komplexitet än i Linåker (2003). Dessa datamängder karakteriseras av kontinuerligt sjunkande eller stigande värden, vilket ger beskrivningar med diffusa eller överlappande gränser i indatarymden.

Studien är uppdelat i två experiment som kompletterar varandra. I det första experimentet testas ARAVQ på en syntetisk datamängd med ett enkelt beroende. Den

syntetiska datamängden antas ge en bild av hur ARAVQ hanterar kontinuerligt stigande och sjunkande värden i isolation från andra påverkande faktorer. I det andra experimentet testas ARAVQ på en fysiologisk datamängd som är tagen från en reell datautvinningsuppgift. Den fysiologiska datamängden beskriver de fysiologiska tillstånden hos ett antal försökspersoner under en begränsad tidsperiod och antas ha en komplexare struktur än den syntetiska datamängden, bland annat ett temporalt beroende.

Det skapas två versioner av den fysiologiska datamängden. Genom att låta datamängderna som används vid utvecklingen av datautvinningsalgoritmerna skilja sig åt olika mycket från de datamängder som används vid testningen av datautvinningsalgoritmerna ges en bild av hur väl lösningarna kan generaliseras till nya situationer.

2 Datautvinning

Organisationer väljer att spara mer och mer data med argumentet att den säkerligen innehåller mycket kunskap, om till exempel kundbeteenden och trender. På grund av den enorma mängd data som sparas är det ofta inte möjligt att analysera den utan hjälpmedel. En ansats för sådana hjälpmedel är datautvinning (eng. Data Mining) (Han & Kamber, 2001).

Enligt Friedman (1997) är det svårt att sätta en tydlig gräns för vad som inkluderas inom datautvinning. Beroende på vem som beskriver det ges olika definitioner och samma definitioner kan ges för olika termer. För detta projekt används definitionen från Han och Kamber (2001):

> ”Data mining is the task of discovering interesting patterns from large amounts of data where the data can be stored in databases, data warehouses, or other information repositories.” (Han & Kamber, 2001, s.33)

Han och Kamber anser att datautvinning går ut på att lyfta fram intressanta mönster ur en stor mängd data som finns lagrad i en databas, ett datalager eller någon annan form av informationskälla. Definitionen inbegriper även de steg som i vissa artiklar beskrivs tillhöra det mer övergripande området ”Knowledge Discovery in Databases” (KDD). KDD-processen innefattar städning och integrering, urval och transformering, datautvinning samt utvärdering och presentation av funna mönster. Han och Kambers definition används för denna studie eftersom algoritmerna inte uteslutande kan kategoriseras till datautvinningssteget i KDD-processen.

Han och Kamber (2001) förtydligar definitionen av datautvinning med att förklara att knappast alla mönster som datautvinningsverktyget hittar är intressanta. Mönstren är intressanta först när de enkelt kan förstås av människor, fungerar på nya datamängder, är användbara samt visar någonting nytt. De kan också vara intressanta om de bekräftar en hypotes som användaren ställt upp. Själva datautvinningen är alltså en del av någonting större, med förberedelser, krav och förväntningar.

Pyle (1999) fortsätter i samma spår och menar att datautvinning är ett steg i en större process som genomförs för att ge underlag till beslutsfattande. Denna process består av fyra steg där datautvinningssteget är det sista och mest tidskrävande steget. Trots detta menar Pyle att de tre första stegen är av betydligt större vikt för att nå ett bra resultat. Dessa tre steg handlar om att hitta ett problem som kan lösas, definiera vad som egentligen menas med ett ”bra resultat” och specificera hur resultatet skall implementeras så att det verkligen ger en förändring i praktiken. Pyle (1999) understryker att datautvinningen knappast är en magisk del av processen, utan att även detta steg kräver mycket förberedelse och kunskap om problemdomänen. Han menar att det inte finns någon generell datautvinningsalgoritm och att det är problemet som skall styra vilka verktyg som används, snarare än att verktyget ska bestämma vilket problem som ska lösas.

2.1 Typ av data inom datautvinning

Mycket av den information som används inom datautvinning kommer från företagens databaser. De kan exempelvis innehålla information om kundernas köpbeteenden, lagerhållning och logistik. Ofta har dessa byggts upp med idén om att informationen är värdefull men utan riktigt vetskap om vad den skall användas till, vilket medför att

den inte är anpassad för datautvinningsuppgiften. Detta är en av orsakerna till att datamängden behöver förberedas (Pyle, 1999). Detta tas upp i avsnitt 2.4.

Informationen inom datautvinning kan också komma från helt andra områden såsom medicin och fysiologi. Den kan lagras i andra former än relationsdatabaser, till exempel på Internets ”World Wide Web”, i textfiler, i spatiala databaser, som komplexa objekt i en objektorienterad databas eller som tidsserier i en temporal databas (Han & Kamber, 2001). Dessa lagringsformer är anpassade för olika typer av datamängder som ställer olika krav på de datautvinningsverktyg som kan användas.

Det som är gemensamt för datamängderna är att de åtminstone i någon grad överensstämmer med verkligheten. Detta är en premiss för att kunna utveckla modeller som fungerar i praktiken. Används en felaktig datamängd kommer även resultatet att bli felaktigt, oberoende av vilket datautvinningsverktyg som används.

En speciellt svårhanterlig, men ändå vanligt förekommande, datatyp inom datautvinning är tidsserier (Pyle, 1999). Dessa består alltid av minst två dimensioner, där förvisso en av dimensionerna - tiden - kan representeras implicit. I tidsserier är ordningen mycket viktig och ställer speciella krav på hur datamängden förbereds. Det är till exempel problematiskt att göra slumpmässiga urval ur en sådan datamängd i och med att det finns risk för att ordningen förstörs.

2.2 Klassificering av data inom datautvinning

Han och Kamber (2001) redogör för två huvudkategorier av uppgifter inom datautvinning. Den första huvudkategorin är beskrivande uppgifter. De beskrivande uppgifterna handlar om att lyfta fram de generella egenskaperna hos en datamängd. Hit räknas bland annat klustring som används dels för att underlätta hantering och översiktlighet av en datamängd direkt för användaren och dels som ett förberedande steg i datautvinningsprocessen (avsnitt 2.4).

Den andra huvudkategorin är förutsägande uppgifter. Dessa handlar om att dra slutsatser från den nuvarande datamängden för att göra förutsägelser om data som inte finns tillgänglig. Hit räknas bland annat klassificering. Med klassificering hittas en mängd modeller som beskriver och skiljer en datamängd i klasser. Dessa klasser används för att göra förutsägelser om nya exempel och datamängder, vilket är användbart vid exempelvis kreditprövning, medicinska diagnoser och selektiv marknadsföring.

Han och Kamber (2001) ser klassificering som en 2-stegsprocess, där det första steget är att skapa en modell som beskriver en förutbestämd mängd av klasser. Modellen kan vara ett beslutsträd eller ett artificiellt neuronnät som tränats på exempel (vanligen representerade av tupler i en databas). Dessa exempel motsvarar en beskrivning av ett objekt eller en händelse, och har ett så kallat klassattribut som beskriver vilken klass det tillhör. Datamängden som används för träningen är en delmängd av den totala tillgängliga datamängden och kallas för träningsmängd.

I nästa steg kontrolleras om modellen presterar ett tillräckligt högt antal korrekta klassificeringar för att var godtagbar. Detta kontrolleras till exempel genom att testa den mot valideringsmängden, vilket är en del av den totala tillgängliga datamängden som inte använts som träningsmängd. Genom att använda två skiljda datamängder för träning och validering undviks att arkitekturen övertränar på datamängden, det vill säga att den lär sig anomalier som inte finns presenterade i den totala tillgängliga datamängden. Om modellen ger ett godtagbart antal korrekta klassificeringar kan den användas för att klassificera kommande datamänger, där klassattributet inte är känt.

2.3 Datautvinningsarkitekturer

En tupel i databassammanhang motsvaras av ett exempel inom datautvinning. Ett exempel representeras av en indatavektor som motsvaras av en punkt i en flerdimensionell rymd. De flesta datautvinningsarkitekturer syftar till att dela upp rymden i områden så att indatavektorerna inom ett visst område har gemensamma egenskaper (Pyle, 1999). Uppdelningen i områden baseras på nuvarande datamängd och kan sedan användas för beskrivningar och förutsägelser av såväl nuvarande som nya datamängder.

Pyle (1999) förklarar att beslutsträd och artificiella neuronnät är två mycket vanliga datautvinningsarkitekturer. Beslutsträd delar in rymden så att områdena skiljer sig maximal från varandra enligt någon utvald egenskap. Han och Kamber (2001) förklarar att beslutsträd är uppbyggt som ett flödesschema, där varje nod motsvarar ett test för ett attributs värde, varje gren motsvarar resultatet av testet och varje löv representerar en klass. Dessa kan sedan uttryckas i form av enkla regler, till exempel ”OM X > 0,5 OCH Y > 0,51 SÅ …”. Områdenas gränser följer parallellt med rymdens dimensioner, vilket gör det svårt att använda beslutsträd på problem som kräver icke-linjära funktioner. Enligt Pyle (1999) passar istället artificiella neuronnät betydligt bättre på denna typ av problem.

Ett artificiellt neuronnät består av en mängd noder med viktade kopplingar mellan varandra (avsnitt 3.2). Det skapar ett matematiskt uttryck som kan ta formen av en icke-linjär funktion, vilket ger fördelen att fler typer av problem kan lösas med ett mer exakt resultat (Pyle, 1999). Andra fördelar med neuronnät är att de inte speciellt bruskänsliga samt att lösningarna kan generaliseras till nya datamängder. Det för även med sig en nackdel, nämligen att det blir betydligt svårare att omvandla funktionen till lättförståeliga regler. (Han & Kamber, 2001).

2.4 Förberedande av data inom datautvinning

Oberoende av vilken datautvinningsarkitektur som används kan den dra fördel av att datamängden förbereds (Pyle, 1999; Han & Kamber, 2001).

Att förbereda datan går ut på att informationen som finns gömd inom den görs så tillgänglig som möjligt för datautvinningsverktyget. Det är viktigt att behålla och lyfta fram datamängdens underliggande egenskaper för att skapa en verkligt fungerande modell. Pyle (1999) hänvisar till att den absolut vanligaste källan för datautvinning är databaser. Ofta är den data som finns lagrad i dessa insamlad för något annat syfte än datautvinning, vilket innebär att den måste förändras innan den används. Exempelvis är enbart vissa av attributen är relevanta, icke-numeriska värden behöver omvandlas till numeriska värden och andra värden behöver läggas på rätt abstraktionsnivå.

Vidare är det mycket vanligt att databaser innehåller brus, saknar värden eller är inkonsekventa (Han & Kamber, 2001). Bruset kommer exempelvis ifrån att mätinstrumenten inte ger helt korrekta värden, mänskliga fel, fel vid överföringen av data eller tekniska begränsningar. Att vissa värden saknas kan bero på att de inte ansågs viktiga eller fanns tillgängliga när datamängden matades in, fel på utrustningen eller att de tagits bort för att de inte stämde överens med andra värden. Inkonsekvent data kan bero på en diskrepans mellan vilka benämningar som används eller på något av de ovanstående exemplen. Får datautvinningsverktygen denna datamängd omodifierad kommer en bristfällig och missvisande modell att skapas (Pyle, 1999). För att öka användbarheten av resultatet är det därför nödvändigt att förbereda

datamängden, vilket bland annat innebär att reducera brus, normalisera värden och att ersätta saknade värden (Pyle, 1999; Han & Kamber, 2001; Reinartz, 2002).

Olika tekniker är olika känsliga för avvikelser i datamängden. Pyle (1999) exemplifierar med att beslutsträd inte alls är lika känsliga för saknade värden som artificiella neuronnät. Artificiella neuronnät kräver att de värden som saknas antingen ersätts av något annat värde (förvalt eller härlett) eller att de exempel som de ingår i tas bort. Alla tillvägagångssätten har sina problem. Att ersätta med ett förvalt värde innebär ett antagande om att alla saknade värden är identiska, vilket antagligen inte är fallet. Att ersätta med ett härlett värde innebär att fabricera ett mönster i datamängden, vilket kommer att påverkar resultatet av datautvinningsprocessen. Att helt ta bort exempel som innehåller saknade värden innebär att man minskar datamängden (i vissa fall väsentligt) och eftersom även dessa exempel innehåller information (kanske specifik för just dessa exempel) minskar man möjligheten till att hitta relevanta mönster.

En mycket vanlig metod för att förbereda en datamängd är klustring. Med klustring kan till exempel bruset minskas och uteliggare identifieras. Den kan också användas för datareduktion genom att lyfta fram relevanta likheter och olikheter mellan exemplen i datamängden och plocka bort irrelevanta delar. På så sätt görs informationen som finns i datamängden mer tillgänglig för verktyget, vilket snabbar upp utvecklingen av modellen och ökar antalet korrekta klassificeringar (Han och Kamber, 2001).

För att underlätta utvecklingen av en klassificeringsmodell förbereds alltså de datamängder som kommer att ligga till grund för den. Förberedelsen innebär bland annat att normalisera värden, ta bort uteliggare och hantera saknade värden. Det innebär också att minska storleken på datamängden och lyfta fram likheter och olikheter genom datareduktion. Det finns flera olika metoder för datareduktion, till exempel kan det utföras genom att ta bort redundanta egenskaper, aggregation eller klustring.

Vid klassificering används ofta klustring som datareduceringsmetod. Klustringen sker som ett föregående steg till klassificeringen och grupperar fenomen som liknar varandra. Reinartz (2002) förklarar att processen kan delas in i tre steg. Det första steget innebär att välja ut exempel från en datamängd, i det andra steget klustras datamängden enligt en specifik klustringsalgoritm och i det tredje steget skapas en prototyp som representerar varje kluster. En prototyp är en ”genomsnittsbild” av de exempel som finns i ett kluster, med genomsnittliga värdena för varje dimension. På detta sätt kan en prototyp representera alla fenomen inom hela gruppen (Han & Kamber, 2001). Klassificeringsarkitekturen behöver sedan enbart lära sig att klassificera denna abstrakta representation av fenomenen, där skillnader i datamängden har lyfts fram.

Reinartz (2002) nämner att datareduceringen inom datautvinning inte är tillräckligt undersökt och att det behövs betydligt mycket mer forskning inom området. Vidare förklarar han att en potentiell kunskapskälla för utvecklingen inom området finns i maskininlärningsdomänen.

3 Mobila robotar

Det finns flera ansatser för att modellera intelligens, med olika filosofiska antaganden och tekniska förutsättningar för systemen[1]. Att modellera intelligens med robotar vilar bland annat på antaganden om att det är nödvändigt att ett system befinner sig i den riktiga världen för att det ska kunna reagera snabbt på stimuli, undvika för stora beräkningskrav och fungera i världar som den inte har fullkomlig kunskap om (Brooks, 1991). Brooks menar också att man genom att implementera systemet som en robot, och låta den interagera med en riktig värld, undviker överförenklingar och att bortse från viktiga delar av interaktionen. Är det något som fattas kommer inte implementationen att fungera.

Givetvis blir indatamängden annorlunda när systemet ska interagera med något annat än enkla bokstavskombinationer, som systemen testats på inom de tidigare ansatserna inom AI. Indatamängden hamnar på en lägre abstraktionsnivå och omfattningen kan öka kraftigt, vilket ställer kontrollarkitekturerna inför nya svårigheter.

3.1 Typ av data inom robotik

Jakobi (1997) förklarar att utvecklingen av mobila robotar går mycket snabbare i en simulator än om en fysisk robot skulle utvecklas i en verklig miljö. Han fortsätter dock med att förklara att det är svårt att simulera allt i den verkliga miljön eftersom det skulle bli beräkningsmässigt krävande, men att om inte allt simuleras så finns det risk för att delar av omgivningen som är viktiga för agentens utveckling försummas. För att lösa detta menar Jakobi att de delar av miljön som är viktiga för robotens utveckling skall modelleras exakt och övriga delar i simuleringen skall varieras. Nolfi och Floreano (2000) kritiserar dock detta förslag och menar att det inte på förhand går att veta vilka delar av omgivningen som är viktiga för robotens utveckling.

I flera experiment sker utvecklingen av robotarnas kontrollarkitektur i en robotsimulator (se exempelvis Meeden, 1996; Ziemke, 2000; Linåker, 2003). Ett exempel på en robotsimulator är ”Yet Another Khepera Simulator” (YAKS)(Carlsson & Ziemke, 2001) som används av Linåker (2003). I YAKS simuleras en eller flera Kheperarobotar[2] som interagerar med en enkel värld bestående av väggar, ljuskällor, målzoner samt stationära och flyttbara objekt. Kontrollarkitekturen tränas i simulatorn och kan sedan testas i simulatorn eller i en verklig Kheperarobot.

I YAKS kommer kontrollarkitekturens indata från de sensorer som finns på roboten, till exempel IR-sensorer och kameratorn, och utdata skickas till motorer som styr robotens hjul. På detta sätt ges roboten en möjlighet att utveckla en kontrollarkitektur som beroende på sensorernas aktivation (indata) ger en viss aktivation för motorerna (utdata) vilket gör att roboten kan interagera med omvärlden.

Den data som når kontrollarkitekturen är alltså ”rå” sensordata som inte modifierats av en mellanhand. Miljöerna som används är dock ofta enkla (se exempelvis Meeden, 1996; Ziemke, 2000; Linåker, 2003) och i enlighet med Jakobi (1997) modelleras endast de bitar som antas vara viktiga för robotens utveckling. Forskningen riktas mot möjligheten att överföra en tränad arkitektur från simulering till verklighet, snarare än att den verklighet som används skall vara realistisk.

---

[1] Se Franklin (1995) och Ziemke (2000) för mer information om ansatser för att modellera intelligens.

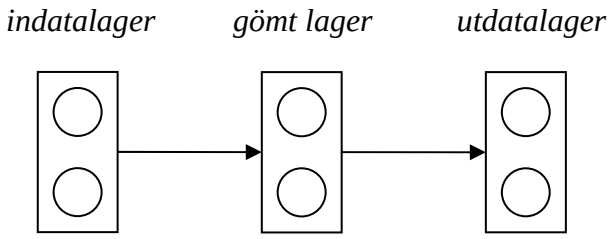
[2] Se Mondada, Franzi och Ienne (1993) för mer information om Kheperaroboten.

Att överföra tekniken till en realistisk situation är sällan en enkel uppgift. Ett exempel på detta är ”DARPA Grand Challenge” (Defense Advanced Research Projects Agency, 2004) som hölls i Kalifornien och Nevada den 13:e mars 2004. Tävlingen gick ut på att en obemannad, helt autonom, bil skulle ta sig från Barstow i Kalifornien till Primm i Nevada, totalt en sträcka på nästan 230 kilometer. Ingen av bilar klarade av att köra längre än 12 kilometer vilket kan antas vara ett misslyckande (Defense Advanced Research Projects Agency, 2004).

3.2 Kontrollarkitekturer för mobila robotar

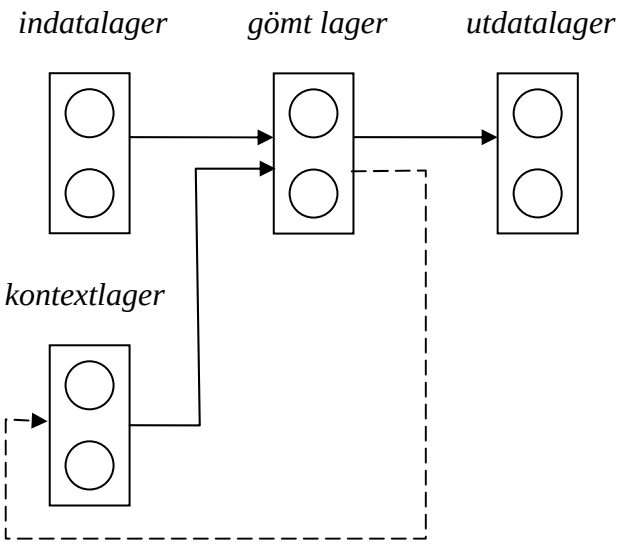
Artificiella neuronnät är enligt Meeden (1996) en av de mest lovande teknikerna för att kontrollera mobila robotar. Med artificiella neuronnät kan indata komma direkt från sensorer och utdata kopplas direkt till motorer, vilket ger en tät koppling mellan perception och handling. Tillsammans med ett synsätt att utveckling skall ske genom anpassning ger dessa möjlighet till att förminska konstruktörens roll och istället låta uppgiften styra utvecklingen av kontrollarkitekturen. Genom denna ”självanpassning” menar Meeden (1996) att systemen självt skapar en lösning samtidigt som det upptäcker de relevanta delarna av problemet, utan att styras av konstruktörens förförståelse av problemdomänen. När systemet har utvecklat ett tillräckligt bra beteende kan det analyseras och ge nya idéer om lösningar på problemet.

Ett artificiellt neuronnät består av flera enkla processande enheter, noder, med kopplingar mellan varandra. Dessa kopplingar har vikter och genom att låta en algoritm uppdatera vikterna kan neuronnätet lära sig att associera indata med utdata (Mehrotra, Mohan & Ranka, 1997). Indatamängden består av en vektor med tal och kan komma från exempelvis sensorer eller en textfil. Utdatamängden består också av en vektor med tal och kan exempelvis styra motorer eller skrivas till en textfil. I sin enklaste form består neuronnätet av en nod vars utdata bestämmer till vilken av två klasser en indatavektor tillhör. Med en nod är det enbart möjligt att göra linjära funktioner. De flesta problem kräver mer komplexa funktioner, vilket kräver att fler noder i fler lager används. Dessa noder är knutna till varandra genom kopplingar. En enkel form av ett artificiellt neuronnät med flera lager är ett enkelt nätverk (eng. feed-forward network), som enbart tillåter kopplingar till nästa lager (Figur 1). En sådan arkitektur som består av två lager (ett gömt lager och ett utdatalager) kan uppskatta vilken icke-linjär funktion som helst (Rumelhart, Widrow & Lehr, 1994).



Figur 1: Artificiellt neuronnät med flera lager, där enbart kopplingar till nästa lager är tillåtna.

Elman (1990) presenterar en arkitektur där det även är tillåtet med kopplingar till tidigare lager, ett enkelt återkopplat nätverk (eng. simple recurrent network) (Figur 2).



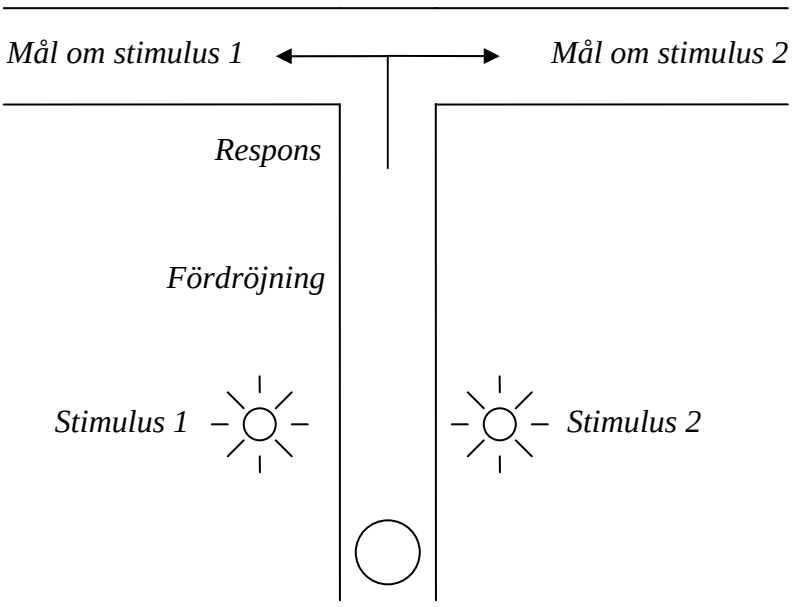
Figur 2: Artificiellt neuronnät med kopplingar till tidigare lager.

Värdet för noderna i det gömda lagret vid tidssteg *t* kopieras till kontextnoderna och presenteras för noderna i det gömda lagret vid tidssteg *t+1*. Denna återkoppling leder till att noderna kan se sin tidigare aktivation vilket ger dem ett minne av tidigare tillstånd

Ett minne av tidigare tillstånd innebär att samma indata kan ge olika utdata beroende på vilken tidigare indata som presenterats för arkitekturen. Kontrollarkitekturen blir alltså kontextkänslig och kan anpassa sin respons till mer än enbart det som presenteras för tillfället.

3.3 Vägmärkesproblemet

Ett exempel på en situation som kräver en kontextkänslig kontrollarkitektur är vägmärkesproblemet (Ryalatt & Czarnecki, 2000). Uppgiften fordrar att roboten ger en fördröjd respons på stimuli. Ryalatt och Czarnecki (2000) förklarar problemet att en robot färdas genom en korridor på väg mot en korsning (Figur 3). Under sin färd mot korsningen observerar den ett vägmärke (stimulus) som snart hamnar utom synhåll för roboten. När roboten når korsningen skall den välja väg (respons) beroende på vilken sorts vägmärke den sett tidigare.



Figur 3: Vägmärkesproblemet. Roboten skall välja väg i korsningen beroende på tidigare stimulus (efter Linåker, 2003).

I och med att roboten inte kan se vägmärket från korsningen behöver den associera indata vid ett visst tidssteg med utdata flera tidssteg senare. Ett idealiskt testfall för ett enkelt återkopplat nätverk.

Linåker (2003) förklarar dock att ett enkelt återkopplat nätverk inte presterar speciellt bra i vägmärkesproblemet. Detta menar han beror på att det är svårt att ställa in kopplingarnas vikter i neuronnätet. Om vikterna för svaga försvinner stimulusaktivationen innan roboten når korsningen. Om vikterna däremot är för starka uppfattar roboten inte ens stimulus när den passerar dem. Vidare menar Linåker att inlärningsalgoritmen som vanligen används, ”back-propagation”, inte passar när det kommer till relationer över lång tid.

Linåker (2003) presenterar en teknik som reducerar indataflödet och enbart ger kontrollarkitekturen indata när det sker en händelse. Förenklat motsvaras en händelse av en förändring i robotens sensorintryck. Dessa händelser har en ordning, till exempel: korridor→lampa på vänster sida→korridor→korsning, men saknar en detaljerad beskrivning av tillståndens utsträckning över tid. Tiden för varje händelse kan variera godtyckligt utan att påverka lösningen på problemet, vilket rejält underlättar för inlärningsalgoritmen.

3.4 Datareduktion för mobila robotar

Linåker (2003) presenterar en teknik för spatial och temporal datareducering för mobila robotar. I exempelvis YAKS får robotens kontrollarkitektur vid varje tidssteg indata från sensorerna i form av en indatavektor fylld med decimaltal. Dessa värden motsvarar aktivation i IR-sensorer, kameratorn et cetera. Indatavektorn beskriver alltså omgivningen vid en viss tidpunkt. När agenten förflyttar sig i en miljö där omgivningen ser likadan ut förblir indatavektorn ungefär densamma (sensorbrus ger dock vissa förändringar).

För vägmärkesproblemet (Figur 3) innebär detta att robotens kontrollarkitektur vid den första sträckan får samma indatavektor vid upprepade tillfällen, nämligen en indatavektor som motsvarar korridor. När roboten färdas framåt i korridoren känner

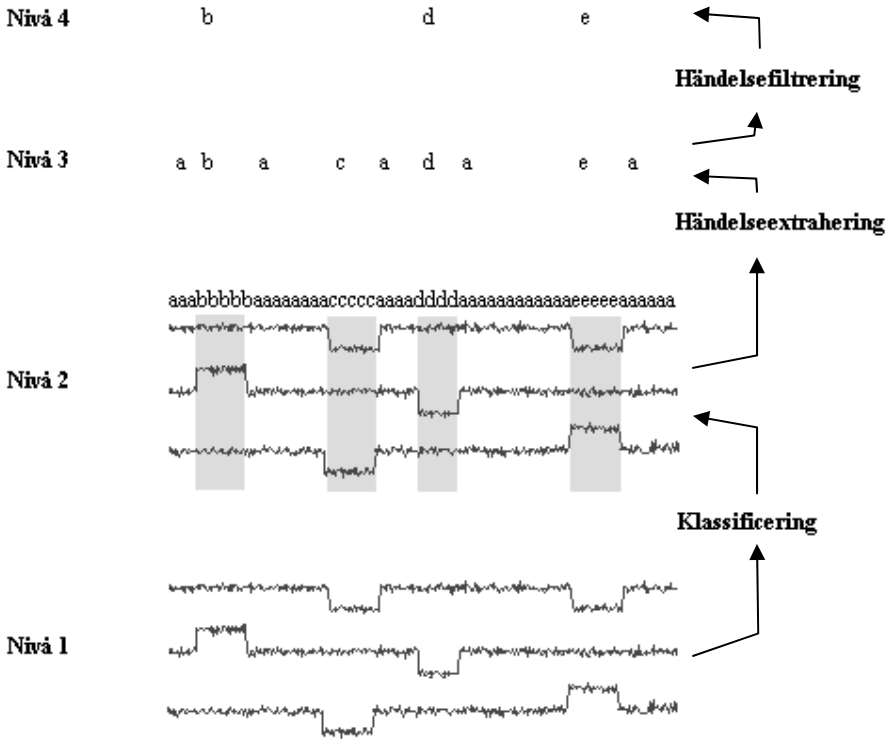
dess sensorer snart av vägmärket, en lampa på vänster eller höger sida av korridoren, vilket således förändrar indatavektorn till att motsvara lampa på vänster sida eller lampa på höger sida. Denna indatavektor upprepar sig till dess att lampan försvunnit utom synhåll för agentens sensorer och förändras sedan till att återigen representera korridor. Samma indatavektor ges till robotens kontrollarkitektur under dess färd genom den sista korridorsträckan innan den når korsningen. Väl framme vid korsningen förändras återigen indatavektorn.

Utan någon form av datareduceringsteknik kommer kontrollarkitekturen att vid varje tidssteg få en ny indatavektor med sensorvärden som indata. För roboten i vägmärkesproblemet kan det innebära 100-talet indatavektorer bara för sträckan fram till korsningen (se exempel i Linåker, 2003). Tekniken som presenteras i Linåker (2003) utför en *temporal datareducering* på indatamängden. Idén är att enbart när indatavektorn förändras tillräckligt mycket skall kontrollarkitekturen underrättas. För den första sträckan i vägmärkesproblemet innebär det att kontrollarkitekturen får indata vid 3 tillfällen: första tillfället den ser korridoren, första tillfället den ser vägmärket, första tillfället den se korridoren (efter att ha sett vägmärket). Över 100 indatavektorer kan alltså sammanfattas till 3 indatavektorer.

Vidare består en indatavektor av värden från varje sensor och motor, ordnade i dimensioner. I YAKS beskrivs aktivationen i en sensor eller motor med ett decimaltal mellan 0 – 1. Med 6 sensorer och 2 motorer består alltså varje indatavektor av 8 dimensioner med decimaltal. Eftersom dessa inte ger exakta värden, utan är påverkade av brus, kommer indatavektorerna att variera mellan varje tidssteg. För att det inte skall uppstå en rad av unika händelser som egentligen motsvarar samma fenomen utför tekniken en *spatial datareducering* med hjälp av *Adaptive Resource Allocating Vector Quantizer* (ARAVQ). När en indatavektor når ARAVQ matchas den mot befintliga prototyper. Är indatavektorn tillräckligt lik någon av prototyperna, det vill säga om den spatialt ligger tillräckligt nära en av prototyperna, ersätts den med värden från prototypen. Prototypen kan dessutom representeras av en vektor med ett annat antal dimensioner, exempelvis en vektor med binära tal där en dimension motsvarar en prototyp. Om flera vektorer representerar samma objekt i omgivningen, men har olika värden på grund av till exempel brus, filtreras de alltså spatialt och ersätts med värdena från en gemensam prototyp. Är indatavektorn inte är tillräckligt lik någon av de existerande prototyperna, skapas en ny prototyp baserad på de senaste tidsstegens indatavektorer.

Tillsammans leder den temporala och spatiala datareduceringen till att kontrollarkitekturen får en betydligt mindre datamängd för inlärning. Genom att enbart ge kontrollarkitekturen indata när det sker en signifikant förändring (benämns som ”händelse” i Linåker (2003)) övervinns inlärningsalgoritmens svårigheter att hantera långvariga temporala beroenden. Linåker (2003) visar att ansatsen framgångsrikt kan användas för vägmärkesproblemet.

Arkitekturen delar in indatamängden i fyra representationsnivåer (Figur 4). Mellan varje nivå modifieras indatamängden. Ju högre representationsnivå, desto större datareducering.



Figur 4: De fyra representationsnivåerna i Linåkers arkitektur (efter Linåker, 2003).

Den *första representationsnivån* motsvarar den råa indatamängden som kommer direkt från den mobila robotens sensorer. Denna är ofta flerdimensionell och ger en kontinuerlig ström av värden. Indatamängden *klassificeras* med ARAVQ, som utför spatial datareduktion, och hamnar i nästa nivå. I den *andra representationsnivån* har varje vektor i indatamängden kopplats till en klass med en beteckning. I Linåker (2003) används bokstäver som beteckning för varje klass, vilket ger att agentens sensoriska intryck under dess färd genom omgivningen kan beskrivas med en rad av bokstäver. Vid *händelseextraheringen* väljs sedan enbart de tillfällen ut när det sker en övergång mellan två olika klasser, det vill säga då det sker en händelse. Till exempel när beteckningen ändras från ”a” till ”b”. Händelserna skickas sedan vidare till nästa nivå. I den *tredje representationsnivån* finns enbart händelser representerade. Av dessa väljs enbart de relevanta händelserna ut genom *händelsefiltrering*. Detta leder till den *fjärde representationsnivån*, där enbart de händelser som anses relevanta finns kvar.

Klassificeringen mellan den första och den andra representationsnivån möjliggör andra sätt att representera indatavektorerna än de som direkt ges av sensorerna. Det här leder till att alla nivåer över den andra representationsnivån kan arbeta på symboliska representationer av omgivningen vilket lättar på systemets beräknings- och lagringskrav.

3.5 Adaptive Resource Allocating Vector Quantizer

ARAVQ används för att klassificera en indatamängd och ersätta de flerdimensionella indatavektorerna med en abstrakt, mer lätthanterlig, beteckning. Vid *tidssteg t* består ARAVQ av en mängd *prototyper P*. Varje prototyp motsvaras av en indatanod, vilket innebär att antalet indatanoder motsvarar mängden *P(t)*. Antalet prototyper är

dynamiskt och ARAVQ skapar nya allteftersom det kommer stabil indata som är tillräckligt olik de existerande prototyperna. Hur många prototyper som skapas beror på komplexiteten i indatamängden samt på hur känslig ARAVQ har ställts in till att vara.

Genom att använda sig av en *indatabuffert X(t)* filtreras brus och ges en bild av de senaste indatavektorerna. Bufferten består av de *n* senaste indatavektorerna (*Ekvation 1*).

$$X(t) = \{x(t), x(t-1), ..., x(t-n+1)\} \qquad \textit{Ekvation 1}$$

Det rörliga medelvärdet $\overline{x}(t)$ baseras på medelvärdet för indatabufferten och räknas ut vid varje tidssteg. Det rörliga medelvärdet är alltså ”filtrerad” indata, som är mindre brusig än de ursprungliga indatavektorerna (*Ekvation 2*).

$$\overline{x}(t) = \frac{1}{n}\sum_{i=0}^{n-1} x(t-i) \qquad \textit{Ekvation 2}$$

ARAVQ startar utan några lagrade prototyper. Dess indatabuffert måste fyllas innan den kan börja med att skapa prototyper och klustra datamängden (*Ekvation 3*).

$$P(n-1) = \varnothing \qquad \textit{Ekvation 3}$$

Nya prototyper läggs till om indatavektorerna i indatabufferten är tillräckligt originella och stabila. För att upptäcka detta används ett avståndsmått $d(V, X)$ som visar det genomsnittligt kortaste avståndet mellan en mängd vektorer (existerande prototyper eller rörligt medelvärde) $v_j \in V$ och en mängd indatavektorer $x_i \in X$. För att mäta avstånd mellan vektorerna används ett mått som baseras på euklidiskt avstånd, betecknas ||.|| (*Ekvation 4*).

$$d(V, X) = \frac{1}{|X|}\sum_{i=1}^{|X|} \min_{1 \le j \le |V|}\{\| x_i - v_j \|\}; x_i \in X, v_j \in V \qquad \textit{Ekvation 4}$$

De nya indatavektorerna anses tillräckligt stabila om den genomsnittliga skillnaden mellan indatavektorerna i indatabufferten och det rörliga medelvärdet (*Ekvation 5*) är lägre än värdet för *stabilitetskriteriet (ε).*

$$d_{\overline{x}(t)} = d(\{\overline{x}(t)\}, X(t)) \qquad \textit{Ekvation 5}$$

Vidare anses de vara tillräckligt originella om de ligger längre ifrån den närmaste prototypen i indatarymden än värdet för *originalitetskriteriet (δ).* Detta avstånd kan inte räknas ut om det inte finns några prototyper lagrade. Istället tilldelas då avståndet ett värde som garanterar att ett nytt kluster skapas (*Ekvation 6*).

$$d_{P(t)} = \begin{cases} d(P(t), X(t)) & |P(t)| > 0 \\ \varepsilon + \delta & |P(t)| = 0 \end{cases} \qquad \textit{Ekvation 6}$$

Om stabilitets- och originalitetskriterierna uppfylls skapas en ny prototyp av det rörliga medelvärdet.

$$P(t+1) = \begin{cases} P(t) \cup \overline{x}(t) & d_{\overline{x}(t)} \le \min(\varepsilon, d_{P(t)} - \delta) \\ P(t) & \textit{annars} \end{cases} \qquad \textit{Ekvation 7}$$

Den nod vars prototyp bäst matchar det rörliga medelvärdet väljs ut till vinnande nod, *win(t)*. Det är denna nod/prototyp som representerar indatavektorn vid tidssteg *t* (*Ekvation 8*).

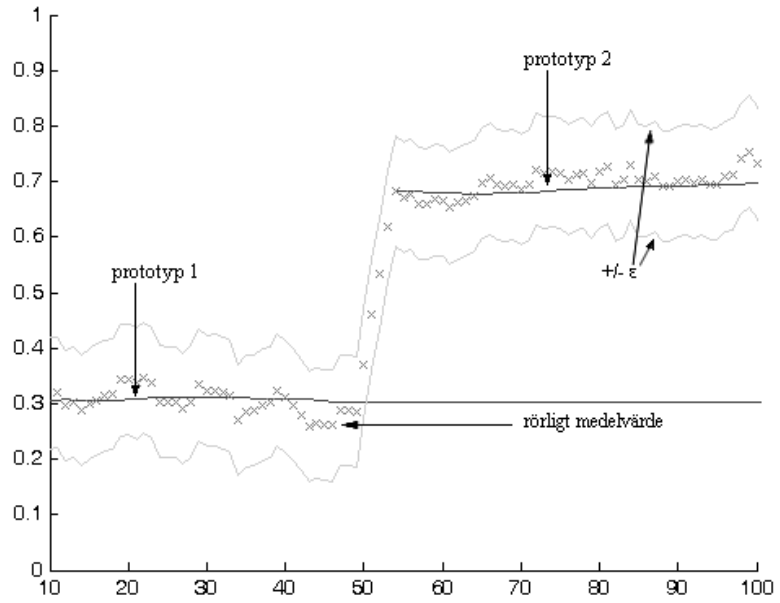
$$win(t) = \arg\min_{1 \le j \le |P(t)|} \{ \| \bar{x}(t) - p_j \| \}; p_j \in P(t) \qquad \textit{Ekvation 8}$$

Om det rörliga medelvärdet ligger tillräckligt nära prototypen för den vinnande noden, anpassas prototypen till det rörliga medelvärdet genom att flyttas närmare detta (*Ekvation 9*). Parametern α påverkar hur mycket prototypen skall anpassas till det rörliga medelvärdet.

$$\Delta p_{win(t)} = \begin{cases} \alpha[\bar{x}(t) - p_{win(t)}] & \| \bar{x}(t) - p_{win(t)} \| < \frac{\varepsilon}{2} \\ 0 & \textit{annars} \end{cases} \qquad \textit{Ekvation 9}$$

Linåker (2003) förklarar att det är stabilitets- och originalitetsparametrarna (ε respektive δ) som har den största inverkan på hur många kluster som ARAVQ skapar. Ökas ε släpps det på stabilitetskravet vilket gör att även kortvarig indata kan skapa ett nytt kluster. Ökas δ ökar kraven på hur mycket ny indata måste skilja sig från befintliga kluster för att ett nytt kluster skall skapas. Om parametrarna ges olämpliga värden ökar risken att för få eller för många kluster skapas. Skapas för få kluster kommer inte klassificeringsverktyget att få tillräckligt med information för att framgångsrikt genomföra sin uppgift. Skapas för många kluster blir själva klustringsprocessen instabil, vilket för att olika kluster kan användas för samma indata vid olika tidpunkter (Linåker, 2003).

Nedan följer ett exempel på hur ARAVQ klustrar en enkel tidsserie. Exemplet är en replikering av ett experiment i Linåker (2003) (Figur 5).



Figur 5: ARAVQ hanterar en enkel tidsserie där värdet förändrats från 0,3 till 0,7 vid tidssteg 50. Värdena är påverkade av +/- 5 % brus. Parametervärdena för ARAVQ är: n=5, α=0,03, ε=0,10 och δ=0,20. (replikering av Linåker, 2003).

Exemplet visar att ARAVQ vid tidssteg 10 har skapat en prototyp med ett värde strax över 0,3. I och med att nya indatavektorer presenteras påverkas det rörliga medelvärdet och prototypen som bäst matchar detta förändras något. När det rörliga medelvärdet skiljer sig mer än ε/2 anpassas inte längre den första prototypen till det rörliga medelvärdet. Vid tidssteg 50 ändras värdet i tidsserien till 0,7 och en ny prototyp skapas vid tidssteg 55. Den andra prototypen anpassas sedan till det rörliga medelvärdet under de resterande tidsstegen. Resultatet efter att ARAVQ klustrat indatamängden är att de tidigare 100 unika indatavektorerna representeras av 2 prototyper som ligger mycket nära de ursprungliga värdena (0,3 respektive 0,7) innan de påverkats av brus.

ARAVQ har visats fungera bra i en enkel robotikmiljö, som är både begränsad och statisk (Linåker, 2003). Trots detta menar Linåker att det är möjligt att generalisera tekniken till andra domäner och han skriver:

> ”In other words, the data reducer could possibly be used as a general time series data mining tool.” (Linåker, 2003, s.179)

Inom datautvinning finns det en stor variation av möjliga strukturer och egenskaper i datamängderna. Dessa kan vara betydligt mer komplexa än den enkla robotikmiljö som används i Linåker (2003).

4 Problemprecisering

Projektets mål är att undersöka lämpligheten av att applicera ARAVQ som datareducerare för att underlätta senare klassificering med artificiella neuronnät. Studien tar sin utgångspunkt från datautvinning och dess behov av fungerande datareduceringstekniker.

Inom datautvinning hanteras mycket stora mängder data i syftet att utvinna kunskap. Med hjälp av tekniker som reducerar datamängdernas storlek förtydligas relevanta mönster, vilket underlättar datautvinningsprocessen. Reinartz (2002) nämner dock att dessa datareduceringstekniker ofta är bristfälliga och att det behövs betydligt mycket mer forskning inom området. Vidare förklarar han att en potentiell kunskapskälla för utvecklingen inom området finns hos maskininlärningsdomänen.

ARAVQ är en datareduceringsteknik som utvecklats för mobila robotar (Linåker, 2003). Tekniken har tidigare framgångsrikt testats i simuleringar där en robot interagerar med en enkel och konstgjord miljö. En egenskap hos dessa miljöer är att indatamängden (sensordatamängden) har tydligt avgränsade beskrivningar av fenomenen i omgivningen. De tilltänkta klustren finns alltså på ett mänskligt intuitivt sätt uppdelade i miljön, färdiga för ARAVQ att hitta. Genom att använda denna typ av indatamängd kan det antas att processen för att skapa relevanta kluster – som kan utnyttjas av kontrollarkitekturen – har underlättats betydligt. De utvecklade teknikernas applicerbarhet på verkliga, ej artificiellt skapade, problem kan därför ifrågasättas.

Linåker (2003) förklarar att ARAVQ kan appliceras på andra domäner än robotikdomänen, där det gäller att upptäcka förändringar hos signaler med kontinuerliga värden. Han förlänger resonemanget med att påstå att ARAVQ eventuellt kan användas som ett generellt datautvinningsverktyg för tidsserier. Frågan är om ARAVQ är lika framgångsrik när man frångår de enkla miljöer som används i Linåker (2003) och istället använder komplexare data, där indatamängden inte har lika tydligt uppdelade beskrivningar.

4.1 Projektets mål

Målet med studien är att undersöka lämpligheten av att applicera ARAVQ som datareducerare för att underlätta senare klassificering med artificiella neuronnät. Undersökningen utförs genom att testa ARAVQ på en datamängd som beskriver distinkta fenomen med diffusa eller överlappande gränser i indatarymden.

Målet delas upp i två delmål. Båda delmålen handlar om att klassificera data med hjälp av artificiella neuronnät. ARAVQ placeras som ett föregående steg till klassificeringsverktyget, för att reducera datamängden genom klustring.

- Det första delmålet är att undersöka hur ARAVQ hanterar syntetisk data. Den syntetiska datamängden konstrueras specifikt för detta experiment, vilket gör det möjligt att systematiskt manipulera beroenden och andra egenskaper hos datamängden. Detta antas ge bättre förutsättningar för att hitta orsaksförklaringar till experimentets resultat än om enbart en icke manipulerbar datamängd skulle användas.
- Det andra delmålet är att undersöka hur ARAVQ hanterar fysiologisk data. Den fysiologiska datamängden kan inte i lika stor grad kontrolleras som den syntetiska datamängden och det är svårt att med säkerhet visa på vilka

beroenden som finns i datamängden. Det antas dock att den fysiologiska datamängden ger en referenspunkt till en realistisk datautvinningssituation, vilket minskar risken för övergeneraliseringar och för stora förenklingar av de miljöer och datamängder som systemen kan komma att möta i en faktisk datautvinningssituation.

Den fysiologiska datamängden delas in i två versioner. I den första versionen kommer exekveringsmängden ha stor variation från tränings- och valideringsmängden. I den andra versionen kommer exekveringsmängden ha liten variation från tränings- och valideringsmängden. Tillsammans skall dessa två versioner ge underlag för en analys av generaliserbarheten hos de kluster som ARAVQ skapar.

4.2 Hypotes

Genom att plocka fram och abstrahera signifikanta förändringar i datamängden antas ARAVQ reducera datamängdens omfång och lyfta fram relevanta likheter och olikheter utan att viktig information förloras. Projektets hypotes är således att när indata beskriver distinkta fenomen med diffusa eller överlappande gränser i indatarymden kommer ett artificiellt neuronnät med ARAVQ som föregående steg att ge lika många eller ett högre antal korrekta klassifikationer än en klassificeringsarkitektur som inte har ARAVQ som föregående steg.

4.3 Begränsningar

Beroende på hur klassificeringsverktygen bearbetar datamängden kan de antagligen dra olika stor nytta av ARAVQ. I studien kommer enbart artificiella neuronnät att användas som klassificeringsverktyg, vilket minskar möjligheten att generalisera resultatet till andra klassificeringsverktyg såsom beslutsträd eller bayesisk klassificering (eng. bayesian classification). Artificiella neuronnät har dock i tidigare experiment med framgång använts tillsammans med ARAVQ (se Linåker, 2003) och antas ge en god utgångspunkt för jämförelser med dessa experiment.

Vidare är ARAVQ den enda datareduceraren som undersöks i studien, vilket innebär att klassificeringsarkitekturen som inte har ARAVQ som föregående steg får ”oklustrad” rådata som indatamängd. Studiens resultat visar hur väl ARAVQ hanterar denna typ av datamängd, snarare än hur väl den står sig i jämförelse med andra typer av datareduceringsalgoritmer.

5 Experiment

Målet med studien är att undersöka lämpligheten av att applicera ARAVQ som datareducerare för att underlätta senare klassificering med artificiella neuronnät. Undersökningen skall utföras genom att testa ARAVQ på datamängder som beskriver distinkta fenomen med diffusa eller överlappande gränser i indatarymden.

I undersökningen utvecklas artificiella neuronnät till att klassificera exempel i en datamängd. Hälften av de artificiella neuronnäten utvecklas med data som förarbetats av ARAVQ och hälften med data som inte förarbetats av ARAVQ. Resultaten för de två olika förutsättningarna jämförs, vilket antas ge en bild av effekterna från användningen av ARAVQ,

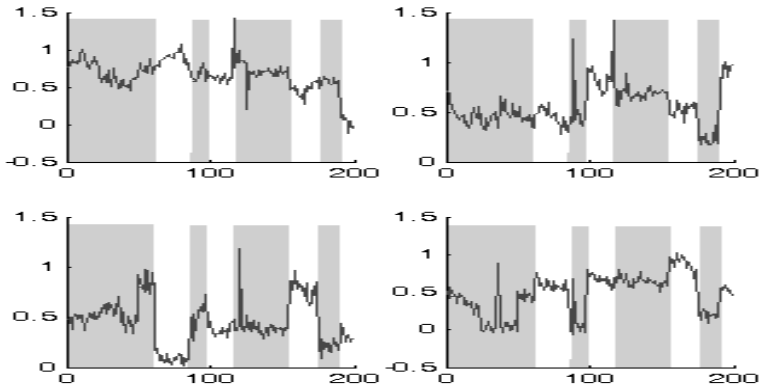
De artificiella neuronnäten kommer att utvecklas till att klassificera exempel i en av två skilda datamängder, en artificiell respektive en fysiologisk datamängd. Datamängderna tas upp i varsitt experiment.

5.1 Datamängder

I det första experimentet används en syntetisk datamängd. Den syntetiska datamängden har skapats specifikt för detta experiment och är i hög grad möjlig att kontrollera och manipulera. I det andra experimentet används en fysiologisk datamängd. Den fysiologiska datamängden innehåller komplexa beroenden och egenskaper som är svåra att kontrollera, något som antas vara vanligt i en praktisk datautvinningssituation.

5.1.1 Syntetisk datamängd

Den syntetiska datamängden ger en bild av hur ARAVQ hanterar kontinuerliga värden när de presenteras i en kontext som saknar andra inverkande faktorer, till exempel andra typer av beroenden och gömda strukturer. De kontinuerliga värdena kan förändras under beskrivning av samma fenomen och beskrivningarna av olika fenomen kan överlappa. Datamängden antas alltså ge möjlighet till att se hur väl ARAVQ klustrar en datamängd med beskrivningar som har diffusa eller överlappande gränser i indatarymden, när de presenteras som en isolerad företeelse. Datamängden består av fyra indatadimensioner som är kopplade till en utdatadimension (Figur 6).



Figur 6: Visar diagram över värdena hos de fyra indatadimensionerna i träningsmängden av den syntetiska datamängden. Värdet för utdatadimensionen visas med hjälp av bakgrundsfärgen: grått område motsvarar värdet 0.95 (tillhör klassen) och vitt område motsvarar värdet 0,05 (tillhör ej klassen).

Utdatadimensionen har värdet 0,05 eller 0,95, vilket räknas ut genom att applicera en godtycklig algoritm på indatadimensionernas värden (Figur 7).

Så länge som antal_skapade_exempel < antal_exempel_som_ska_skapas

Skapa ett exempel och slumpa ut värden mellan 0 och 1 för dess fyra dimensioner

Beräkna exemplets klass med hjälp av algoritmen:

$$\text{klassvärde} = \frac{exempel[1]^2}{exempel[2]} * \frac{exempel[3]}{exempel[4]}$$

om klassvärde > 0.5
    klass = 0.95
annars
    klass = 0.05

Så länge som antal_skapade_exempel_för_serien < antal_exempel_som_ska_skapas_för_serien

Skapa ett nytt exempel där varje dimension antar samma värden som tidigare exempel

Multiplicera värdena för varje dimension i det nya exemplet med 1,05 eller 0,95 (slumpas fram)

Avsluta loopen för att skapa exempel för en serie

Lägg +/- 5 % brus på alla exempel och dimensioner

Lägg till uteliggare med 1 % sannolikhet för varje exempel. Uteliggarna får sina värden för alla fyra dimensioner modifierat med ett brus på +/- 50 %

Avsluta loopen för att skapa exempel

Figur 7: Pseudokod för att skapa den artificiella datamängden.

Indatadimensionerna har värden mellan 0 – 1 som ligger i serier om 5 – 50 värden, där alla exempel i en och samma serie beskriver ett och samma fenomen. I och med att den artificiella datamängden innehåller serier kan indatabufferten i ARAVQ tilldelas ett högre värde än n=1. Hade däremot inte serier använts skulle lämplig parameterinställning vara n=1. Enligt Linåker (2003) innebär inställningen n=1 att ARAVQ fungerar som en enkel ”leader algorithm”, vilket plockar bort fördelarna med ARAVQ i jämförelse med andra datareduceringstekniker. ARAVQ är alltså konstruerad för datamängder som innehåller serier och testas därför på denna typ av data.

För varje serie slumpas det ut antalet värden som skall finnas i serien samt seriens startvärde. Varje värde i serien utgår från det tidigare värdet med en liten ökning eller minskning, med utgångspunkt från startvärdet. Ovanpå de genererade värdena läggs sedan 5 % brus och slumpvis valda värden tilldelas extremvärden (uteliggare). Totalt

skapades 382 exempel med värdet 0,05 (ej tillhörande klassen) respektive 637 exempel med värdet 0,95 (tillhörande klassen) för utdatadimensionen.

I och med att indatadimensionerna kan anta alla värden inom intervallet 0 – 1 hamnar beskrivningarna av fenomenen nära varandra i indatarymden. De närliggande beskrivningarna skapar tillsammans med bruset situationer där fenomenen överlappar i indatarymden. Avgränsningarna mellan fenomenen är alltså inte lika tydliga som i Linåker (2003), där en vägg motsvarar värdet 1 (plus brus) och ingen vägg motsvarar värdet 0 (plus brus).

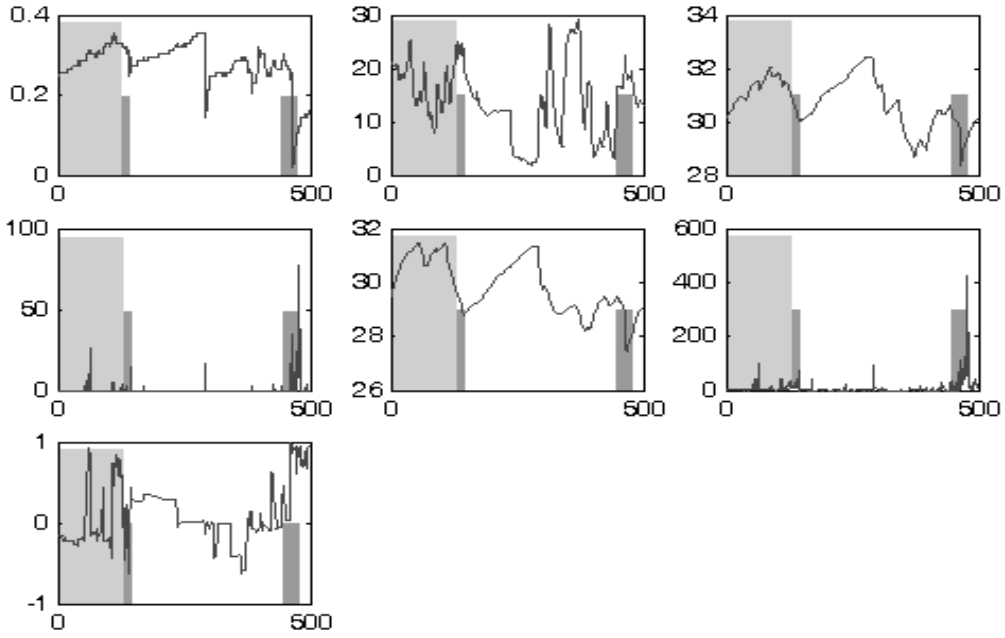
5.1.2 Fysiologisk datamängd

Den fysiologiska datamängden kommer från Physiological Data Modeling Contest (PDMC). PDMC är en tävling som anordnats i samband med en workshop för att få forskarna inom maskininlärning intresserade av fysiologisk data.

Datamängden är insamlad med hjälp av ett speciellt armband (BodyMedia SenseWear Pro Armband) som burits av 18 försökspersoner. Det beskriver deras fysiologiska tillstånd, såsom värmeutveckling och svettningar, vid en serie av tidpunkter (totalt 580 264 tillfällen). Datamängden innehåller för varje tidpunkt värden för två okända egenskaper, nio sensorvärden, försökspersonens kön samt vilken aktivitet försökspersonen ägnar sig åt. Målet i PDMC är att med hjälp av en maskininlärningsteknik klassificera försökspersonernas kön samt om de utövar någon av två aktiviteter i en ny datamängd som saknar denna information. Det handlar alltså om att hitta mönster i datamängden för att kunna göra förutsägelser om den.

Tidpunkterna för mätningarna bildar en serie av sammanhängande värden (Figur 8). Dessa värden förändras inte bara i övergången till beskrivningen av ett nytt fenomen utan även under den tid som datamängden beskriver ett och samma fenomen. Distinktionen mellan två olika fenomen är inte heller lika tydlig som i Linåker (2003), utan beskrivningarna ligger nära varandra i indatarymden.

Det antas att de fysiologiska värdena vid tidpunkt *n* utgår från de fysiologiska värdena vid tidpunkt *n-1, n-2…* och att klassificeringen av nuvarande tillstånd är beroende av vilka tidigare tillstånd personen befunnit sig i. Detta härleds från att en person som springer vid någon tidpunkt kan ha samma fysiologiska värden som en person som cyklar (till exempel lika hög puls), men att förändringen under de senaste tidsstegen skiljer sig åt. Datamängden antas alltså innehålla temporala beroenden.



Figur 8: Diagram över nio sensorvärden hos en försöksperson vid en serie av 500 mättillfällen. Värdet för utdatadimensionen visas med hjälp av bakgrundsfärgen: ljusgrått område motsvarar värdet 0.95 (tillhör klassen), vitt område motsvarar värdet 0,05 (tillhör ej klassen) och mörkgrått område motsvarar värdet 0,5 (okänd klasstillhörighet).

Den fysiologiska datamängden antas liksom den syntetiska datamängden beskriva fenomen med diffusa eller överlappande gränser i indatarymden. Utöver detta beskriver den fysiologiska datamängden en situation som eventuellt inkluderar obekanta beroenden och påverkande faktorer. Dessa förutsättningar antas öka möjligheten att generalisera studiens resultat till reella datautvinningssituationer där datamängderna kan innehålla komplexa beroenden och strukturer, i jämförelse med vad som vore möjligt om det enbart användes en syntetisk datamängd.

Följaktligen kompletterar datamängderna varandra och kan resultera i en indikation på problem med användningen av någon av datamängderna, exempelvis att den syntetiska datamängden är alldeles för anpassad till tekniken eller att den fysiologiska datamängden har för komplex struktur för att hanteras av ARAVQ-implementationen i detta experiment.

5.1.3 Förberedelse av datamängder

Att förbereda datamängden innan datautvinningstekniken appliceras är enligt Pyle (1999), Han och Kamber (2001) samt Reinartz (2002) ett mycket viktigt steg i datautvinningsprocessen, eftersom det antas avsevärt öka möjligheterna till att nå ett bra resultat (avsnitt 2.4). För att öka studiens generaliserbarhet till andra datautvinningssituationer kommer därför den fysiologiska datamängden att delvis förberedas (avsnitt 5.5.1).

Den syntetiska datamängden behöver knappast förberedas i och med att den konstruerats med experimentet i åtanke och har värden som är anpassade för klassificeringsarkitekturen. Både den fysiologiska och den syntetiska datamängden delas dock upp i tre delmängder vardera: tränings-, valideringss- samt exekveringsmängd. Tränings- och valideringsmängden används tillsammans under

utvecklingen av ett artificiellt neuronnät. Träningsmängden används för att uppdatera vikterna i neuronnätet och valideringsmängden används för att undvika att neuronnätet lär sig brus (genom att vid varje uppdatering av vikterna i neuronnätet kontrollera att det presterar lika bra eller bättre på valideringsmängden som innan viktuppdateringen). Exekveringsmängden består av tidigare osedda exempel och används för att kontrollera det färdigutvecklade neuronnätets prestanda. Detta innebär att klassificeringsarkitekturen måste generalisera sina lösningar till nya situationer. Generaliserbarhet är en grundläggande egenskap hos artificiella neuronnät och en nödvändighet när det kommer till klassificering inom datautvinning. Utan möjlighet till generalisering kan klassificeringsarkitekturen enbart användas på tidigare sedda exempel, och för dessa är svaret/klassen redan känd.

5.2 Klassificeringsarkitekturer

Som klassificeringsarkitekturer i experimentet kommer ett enkelt nätverk (eng. feed-forward network) samt ett enkelt återkopplat nätverk (eng. simple recurrent network) att användas (se avsnitt 3.2). I ett enkelt nätverk får kopplingarna mellan noderna enbart gå till ett senare lager. Trots sin enkelhet har arkitekturen visats effektiv och är mycket vanligt förekommande inom experiment med artificiella neuronnät (Mehrotra m.fl., 1997).

En egenskap som ett enkelt nätverk saknar är kontextkänslighet. Eftersom ett enkelt nätverk enbart har kopplingar till ett senare lager kan den bara ta hänsyn till den indata som finns för tillfället. Den fysiologiska datamängden antas ha temporala beroenden och dessa leder till att klassificeringen av nuvarande tillstånd är beroende av vilka tidigare tillstånd personen befunnit sig i. Detta innebär att klassificeringsarkitekturen bör ha någon form av minne av tidigare indata. Med hjälp av ett minne kan arkitekturens kommande utdata både bero på den indata som presenteras för tillfället samt den indata som presenterats vid tidigare tillfällen. Elman (1990) presenterar en nätverksarkitektur med dessa egenskaper, ett enkelt återkopplat nätverk. I och med att det i ett enkelt återkopplat nätverk är tillåtet med kopplingar tillbaka till tidigare lager ges de gömda noderna en möjlighet att se sin tidigare aktivation och anpassa kommande aktivation efter denna.

Det finns även andra typer av nätverksarkitekturer som i tidigare studier har visats speciellt lämpade för data med temporala beroenden, till exempel ”Long Short-Term Memory” (Hochreiter & Schmidhuber, 1997). Olika typer av nätverksarkitekturer kan eventuellt dra olika fördelar av ARAVQ beroende på hur de organiserar de interna tillstånden. Detta leder till att ARAVQ:s för- och nackdelar för en viss nätverksarkitektur inte nödvändigtvis är generaliserbara till andra typer av nätverksarkitekturer. Enkelt återkopplat nätverk har dock använts i flera tidigare experiment (se exempelvis Elman, 1990; Linåker, 2003; Ziemke, 2000; Meeden, 1996) och antas ge en god utgångspunkt för att visa på möjligheterna med ARAVQ. Som klassificeringsarkitektur väljs därför enkelt nätverk samt enkelt återkopplat nätverk.

5.3 Implementering

Av den 3-stegsarkitektur som tas upp i Linåker (2003) (avsnitt 3.4) implementeras det första steget, det vill säga det steg som klassificerar indatamängden med hjälp av ARAVQ. ARAVQ implementeras i Matlab version 6.5.1 (The Mathworks, 2004) och valideras genom att replikera ett delexperiment i Linåker (2003). I valideringen testas ARAVQ på att klustra 1-dimensionell data. Resultatet från valideringen visas i Figur

5. För de artificiella neuronnäten används befintlig implementation i ”Matlab - Neural Network Toolbox” version 4.0.2.

5.3.1 Representation av kluster

När ARAVQ klustrat indatamängden ersätts varje exempel med en ny vektor. Den nya vektorn har lika många dimensioner som antalet kluster ARAVQ skapat. En dimension motsvarar ett kluster och den dimension/det kluster som bäst överensstämmer med exemplet tilldelas värdet 1. Övriga dimensioner tilldelas värdet 0. Till exempel om ARAVQ har skapat 5 kluster kommer en indatavektor som bäst överensstämmer med prototypen för det andra klustret att representeras av vektorn [0 1 0 0 0]. Dessa nya vektorer används sedan istället för originalvektorerna som indatamängd till det artificiella neuronnätet.

I det artificiella neuronnätet motsvaras en dimension av en indatanod, vilket alltså innebär att ett kluster motsvarar en indatanod. Denna typ av lokala kodning (eng. localistic encoding) används av Linåker (2003). Han nämner dock att ansatsen leder till att information förloras eftersom varje kluster hamnar på exakt samma avstånd oberoende av hur lika varandra de är. Vidare nämns att det kan vara problematiskt att lägga till kluster i efterhand eftersom det i så fall krävs att de andra delarna av arkitekturen kan ta emot ett dynamiskt antal dimensioner/kluster. Linåker (2003) kommer runt problemet genom att använda en begränsad och statisk miljö, där systemet konvergerar mot ett visst antal kluster efter att befunnit sig i miljön under en tid.

I denna studie kommer ARAVQ-implementationen och de bakomliggande antagandena så långt som möjligt överensstämma med det som används i Linåker (2003). Detta innebär bland annat att samma typ av representation av klustren används.

5.3.2 Parametrar för ARAVQ

Eftersom det är två olika datamängder som används, en artificiell respektive en fysiologisk datamängd, med olika antal dimensioner, exempel, spridningar et cetera, kommer parameterinställningar för ARAVQ att skiljas åt mellan de båda experimenten. Parameterinställningarna väljs utifrån de värden som används i Linåker (2003).

5.4 Experiment 1 – syntetisk datamängd

I experiment 1 utvecklas de artificiella neuronnäten till att klassificera den syntetiska datamängden. Den syntetiska datamängden har skapats specifikt för detta experiment och är en tidsserie med ett enkelt beroende, nämligen samband mellan indatamängden och klasstillhörighet. Datamängden antas ge en bild av hur ARAVQ hanterar diffusa eller överlappande beskrivningar i indatarymden, i en enkel kontext som saknar andra typer av beroenden.

5.4.1 Parametrar för experiment 1

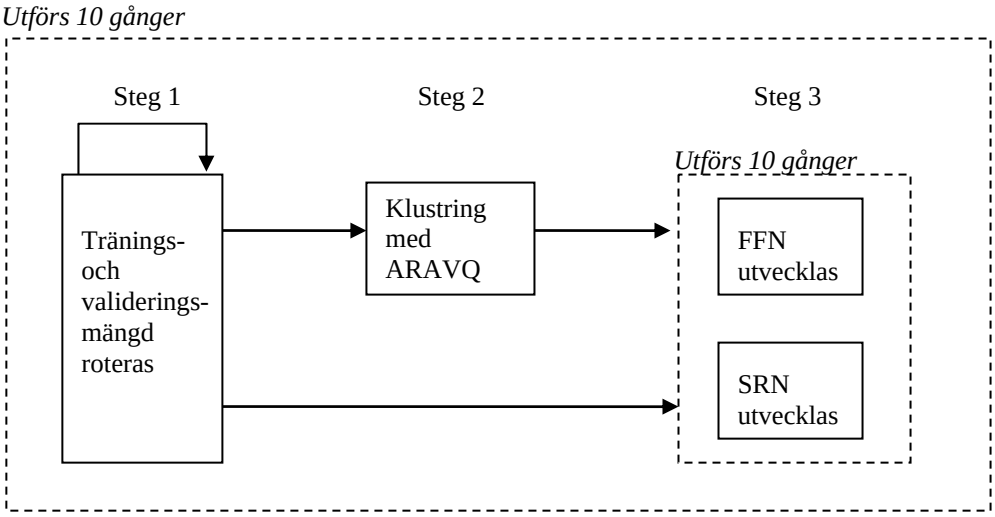
Efter initiala experiment väljs parametervärdena stabilitet (ε) = 0,1 och originalitet (δ) = 0,2, vilket är samma värden som används i ett av experimenten i Linåker (2003, s.62). Tillsammans leder dessa till att ARAVQ skapar 17 kluster av den syntetiska datamängden. Noduppsättningen för det enkla nätverket och det enkla återkopplade nätverket med ARAVQ är 17 indatanoder, 3 gömda noder samt 1 utdatanod.

Noduppsättningen för det enkla nätverket och det enkla återkopplade nätverket utan ARAVQ är 4 indatanoder, 3 gömda noder samt 1 utdatanod.

5.4.2 Upplägg av experiment 1

I experiment 1 kommer det att användas två till en början identiska tränings- och valideringsmängder. Den ena tränings- och valideringsmängden klustras först av ARAVQ innan den används för att utveckla ett enkelt nätverk respektive ett enkelt återkopplat nätverk till att klassificera datamängden. Den andra tränings- och valideringsmängden används obehandlad för att utveckla neuronnäten.

Experiment 1 delas in i 3 steg (Figur 9):



Figur 9: Upplägg för experiment 1.

Steg 1: Träningsmängden roteras. ARAVQ kan skapa olika kluster beroende på i vilken ordning som exemplen presenteras för den. Utvecklingen av klassificeringsarkitekturen kan i sin tur påverkas av vilka kluster som finns tillgängliga. Träningsmängden kommer därför att roteras, den första tiondelen läggs sist i träningsmängden.

Steg 2: *Tränings- och valideringsmängden klustras.* En av de två identiska tränings- och valideringsmängderna klustras av ARAVQ. Varje exempel i originalträningsmängden ersätts med en representation av det bäst matchande klustret. Den andra tränings- och valideringsmängden bearbetas inte av ARAVQ.

Steg 3: *Klassificeringsarkitekturerna utvecklas.* 10 enkla nätverk samt 10 enkla återkopplade nätverk utvecklas per träningsmängd (klustrad respektive icke klustrad träningsmängd). Varje utvecklad klassificeringsarkitektur testas på den tidigare osedda exekveringsmängden (avsnitt 5.4.3).

Återgår till steg 1. Detta görs om färre än 10 rotationer av tränings- och valideringsmängden skett.

Totalt utvecklas det 400 artificiella neuronnät under experiment 1; 10 artificiella neuronnät per arkitektur (2), träningsmängd (2) och rotation (10).

5.4.3 Utvärdering av experiment 1

Varje utvecklat neuronnät testas (steg 3) genom att klassificera exemplen i exekveringsmängden. Under testningen mäts klassificeringsarkitekturernas prestanda med hjälp av måttet e*xakthet (EX),* vilket beskriver den totala andelen korrekta klassifikationer (*Ekvation 10*).

$$EX = \frac{antal\_korrekta\_klassifikationer}{totalt\_antal\_klassifikationer} \qquad \textit{Ekvation 10}$$

För att ge en beskrivning av vilken typ av klassifikationer som arkitekturen har svårast för kompletteras exakthet med 2 andra mått: *känslighet (KÄ)*(eng. sensitivity) och *noggrannhet (NO)* (eng. specificity). Dessa två mått baseras på 4 värden:

- *Sant positiva (SP):* beskriver antalet positiva exempel (exempel som tillhör klassen) som klassificeras som positiva.
- *Falskt negativa (FN):* beskriver antalet positiva exempel som klassificerats som negativa (icke tillhörande klassen).
- *Sant negativa (SN):* beskriver antalet negativa exempel som klassificerats som negativa.
- *Falskt positiva (FP):* beskriver antalet negativa exempel som klassificerats som positiva.

Känslighet beskriver andelen positiva exempel som blir klassificerade som positiva och räknas ut genom att dividera antalet sant positiva klassifikationer med det faktiska antalet positiva exempel (*Ekvation 11*).

$$KÄ = SP/(SP + FN) \qquad \textit{Ekvation 11}$$

Noggrannhet beskriver andelen negativa exempel som blir klassificerade som negativa och räknas ut genom att dividera antalet sant negativa klassifikationer med det faktiska antalet negativa exempel (*Ekvation 12*).

$$NO = SN/(SN + FP) \qquad \textit{Ekvation 12}$$

Resultatet från varje testat neuronnät sammanställs och det aritmetiska medelvärdet räknas ut för varje kombination av nätverksarkitektur (enkelt nätverk respektive enkelt återkopplat nätverk) och datamängd (klustrad av ARAVQ respektive ej klustrad av ARAVQ)(*Ekvation 13*). Detta ger resultat för 4 olika kombinationer.

$$\bar{x} = \frac{1}{n}\left(x_1 + x_2 + ... + x_n\right) \qquad \textit{Ekvation 13}$$

n motsvarar antalet nätverk per kombination av nätverksarkitektur och datamängd. *x* motsvarar exakthet, känslighet eller noggrannhet för ett utvecklat artificiellt neuronnät.

5.5 Experiment 2 – fysiologisk datamängd

I experiment 2 utvecklas de artificiella neuronnäten till att klassificera den fysiologiska datamängden. Den fysiologiska datamängden antas innehålla komplexa beroenden och egenskaper som är svåra att kontrollera. Detta motsvarar i högre grad

de datamängder som används inom praktisk datautvinning, än vad den syntetiska datamängden gör.

Liksom i experiment 1 kommer artificiella neuronnät att utvecklas till att klassificera exempel i en datamängd. Eftersom den fysiologiska datamängden, till skillnad från den syntetiska datamängden, inte är konstruerad specifikt för detta experiment krävs det dock ett par extra steg i experiment 2. Det första steget handlar om att förbereda datamängden så att den enbart består av värden som kan hanteras av ARAVQ och de artificiella neuronnäten. Det andra steget handlar om urvalet. Den fysiologiska datamängden antas innehålla okända temporala beroenden, vilket problematiserar ett slumpmässigt urval på exempelnivå. Vidare innehåller den fysiologiska datamängden ett mycket stort antal NULL-värden för attributet klass vilket försvårar urval och användning av dessa exempel.

5.5.1 Förberedelse av fysiologisk indatamängd

Den fysiologiska datamängden är i sin ursprungsform (när den hämtas från PDMC) redan delvis förberedd: det är bara en av dimensionerna som innehåller saknade värden, nästan alla värden är numeriska och värdena är på samma abstraktionsnivå.

Det är måldimensionen som innehåller saknade värden, vilket innebär att vissa exempel saknar känd klasstillhörighet. Ett exempel som saknar känd klasstillhörighet tillför i sig ingen information för utvecklingen av klassificeringsarkitekturen. Däremot bestämmer ett sådant exempel det interna tillståndet inför kommande klassificeringar och kan vara relevant eftersom datamängden innehåller temporala beroenden. Att ersätta varje saknat klassattribut med ett förvalt värde förmodas försvåra processen att hitta mönster i datamängden, och att härleda de saknade värden med hjälp av andra värden är i sig en åtminstone lika stor uppgift som klassificeringsuppgiften. Istället väljs att plocka bort de exempel som saknar värden för klassattributet. Eftersom experimentet fokuserar på att jämföra arkitekturerna med varandra och att de får samma förutsättningar (förutom datamängden som kan vara klustrad av ARAVQ eller inte klustrad av ARAVQ) antas att resultatet inte påverkas av detta.

Måldimensionen innehåller också värden på en nominalskala, vilket innebär att värdena består av klasser som inte kan rangordnas. Pyle (1999) nämner att detta inte vore något problem att hantera för ett beslutsträd, men att ett artificiellt neuronnät kräver numeriska värden. Om värdena ”direktöversätts” till numeriska värden, det vill säga att klass 3004 översätts till talet 3004, införs en konstlad ordning. Det innebär att exempel som tillhör klass 3004 antas ha större likhet med exempel som tillhör klass 3005, än med exempel som tillhör klass 5102. Detta innebär antagligen att en onödig förskjutning läggs in i datamängden.

För att undvika att en ordning införs kan attributet delas upp i flera dimensioner, en dimension för varje klass. För varje exempel fylls sedan alla dimensioner med talet 0, förutom för den dimension som representerar exemplets klass som fylls med talet 1. I PDMC ska deltagarna förutsäga om ett exempel tillhör klass 3004, klass 5102 samt kön. En så pass stor uppgift antas inte vara nödvändig för denna studie och uppgiften minskas därför ned till att enbart gälla förutsägelse av klass 3004. Övriga klasser är irrelevanta. Därför behålls enbart den dimension som representerar klass 3004.

Avslutningsvis normaliseras värdena i den fysiologiska datamängden till tal mellan -1 och 1.

ARAVQ fungerar som datareducerare, hanterar brus och hanterar olika datadistributioner. Eftersom experimentet går ut på att undersöka möjligheterna med

ARAVQ kommer följaktligen ingen av datamängderna att förberedas genom datareducering, brushantering eller redistribuering. Experimentet kan alltså ses som att inkludera en bit av förberedelsen av datamängderna i och med applicerandet av ARAVQ.

5.5.2 Urval ur fysiologisk indatamängd

Andelen exempel som tillhör klass 3004 är enbart 1 % av det totala antalet exempel. Genomförs utvecklingen med denna fördelning kommer en klassificeringsarkitektur som ger ett negativt svar (utdata 0, det vill säga att exemplet inte tillhör klass 3004) på alla exempel att få 99 % exakthet. Utvecklingen hamnar i ett lokalt optima som kan vara svårt att ta sig ur.

För att undvika detta väljs att enbart en del av exemplen skall finnas kvar. Datamängden förändras så att den ena hälften av mängden består av exempel som tillhör klass 3004 och den andra hälften av mängden består av exempel som inte tillhör klass 3004.

Eftersom datamängden består av en serie där ordningen är en viktig faktor, är det inte möjligt med slumpmässigt urval på en ”exempelnivå”. Klassificeringen av ett enskilt exempel antas vara beroende av vilka tidigare exempel som presenterats (ett temporalt beroende). För att komma runt detta problem väljs att enbart använda exempel från personer som har en relativt sett stor andel exempel tillhörande klass 3004, totalt 4 personer. Från varje person plockas det sedan ut serierna med exempel som tillhör klass 3004 tillsammans med de närmast belägna exemplen.

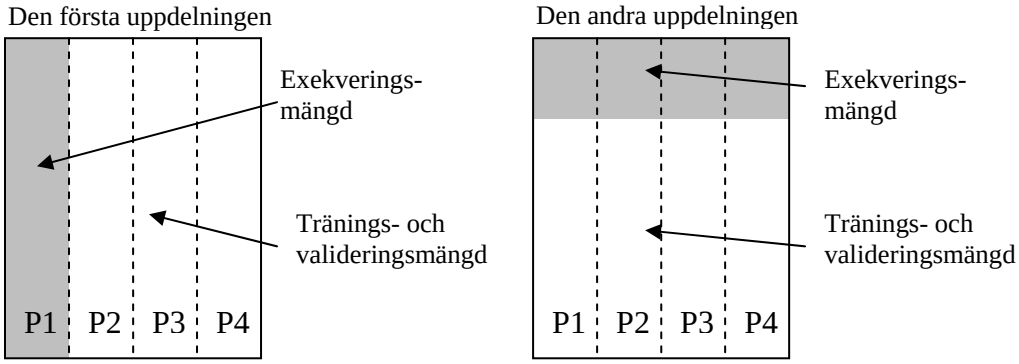
5.5.3 Parametrar för experiment 2

Initiala tester visar på att värdena $\varepsilon = 0.7$ respektive $\delta = 0.2$ ger ett rimligt antal kluster, 46 kluster, för den fysiologiska datamängden. Dessa parameterinställningar används också i ett av experimenten i Linåker (2003, s.153)

De artificiella neuronnäten består av olika antal indatanoder beroende på vilken datamängd som skall hanteras. För datamängderna som inte har förarbetats av ARAVQ kommer 11 indatanoder, 10 gömda noder samt 1 utdatanod att användas. För datamängderna som har förarbetats av ARAVQ motsvaras antalet indatanoder av antalet kluster som skapats av ARAVQ, det vill säga 46 noder. Antalet gömda noder är 10 och antalet utdatanoder är lika med 1.

5.5.4 Upplägg av experiment 2

I experiment 2 kommer det att användas två olika uppdelningar av tränings-, validerings- och exekveringsmängderna. De två uppdelningarna skiljer sig från varandra i urvalet av exempel (Figur 10).

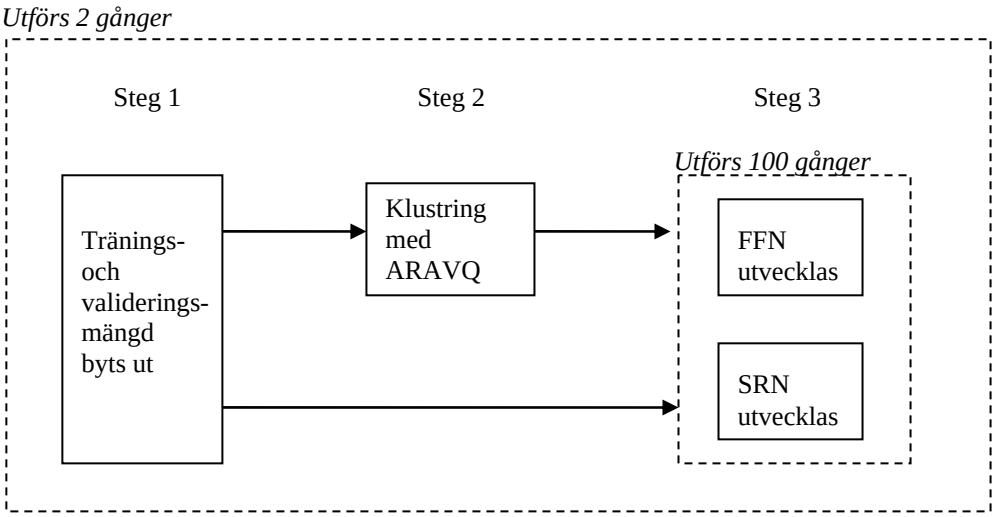


Figur 10: Urvalet av exempel till de två olika uppdelningarna av tränings-, valideringssamt exekveringsmängden. Pn motsvarar mängden exempel som tillhör person n.

I den första uppdelningen består exekveringsmängden av alla exempel hos en och samma person, vilket innebär att tränings- och valideringsmängden består av alla exempel hos de tre övriga personerna. I den andra uppdelningen består exekveringsmängden av ¼ av exemplen hos den första personen, ¼ av exemplen hos den andra personen och så vidare, vilket innebär att tränings- och valideringsmängden består av ¾ av alla exempel från alla personer.

Det antas att den första uppdelningen kräver större generaliserbarhet hos klassificeringsarkitekturen än vad den andra uppdelningen gör. Detta baseras på antagandet om att skillnaden mellan exemplen som används för träning och testning är större i den första exekveringsmängden än i den andra exekveringsmängden.

Experiment 1 delas in i 3 steg (Figur 11):



Figur 11: Upplägg för experiment 2.

Steg1: Tränings-, validerings- och exekveringsmängden byts ut. För att kontrollera generaliserbarheten hos de kluster ARAVQ skapar används två olika datamängder.

Steg 2: *Träningsmängden klustras.* En av de två identiska träningsmängderna klustras av ARAVQ. Varje exempel i originalträningsmängden ersätts med en representation av det bäst matchande klustret. Den andra träningsmängden bearbetas inte av ARAVQ.

Steg 3: *Klassificeringsarkitekturerna utvecklas.* 100 enkla nätverk samt 100 enkla återkopplade nätverk utvecklas per träningsmängd (klustrad respektive icke klustrad träningsmängd). Neuronnäten utvecklas med hjälp av korsvalidering, där delmängderna hos tränings- och valideringsmängden växlar mellan utvecklingsfaserna. Det bästa artificiella neuronnätet från båda arkitekturerna sparas och testas på exekveringsmängden (avsnitt 5.5.5).

Återgår till steg 1. Detta görs om enbart den ena uppdelningen av tränings-, valideringsmängden och exekveringsmängden använts.

5.5.5 Utvärdering av experiment 2

Efter utvecklingen av totalt 800 artificiella neuronnät (2 exekveringsmängder * 2 datamängder * 2 nätverksarkitekturer * 100 artificiella neuronnät), väljs det som har flest korrekt klassificerade exempel (högst exakthet) på valideringsmängderna för varje teknik. Dessa klassificeringsarkitekturer testas sedan på exekveringsmängden.

I experiment 2 används alltså enbart de artificiella neuronnät som presterar bäst för varje arkitektur, datamängd och exekveringsmängd. I jämförelse med medelvärdet som används i experiment 1 kan detta i högre grad jämföras med valet i av klassificeringsarkitektur som sker en reell datautvinningssituation. Detta baseras på antagandet om alla utvecklade arkitekturer vanligtvis inte används vid en faktisk datautvinningssituation, utan att enbart den eller de som presterar bäst.

Vidare roteras inte datamängden innan den klustras av ARAVQ. Detta innebär att klustringen eventuellt skulle ha förbättrats med en annan ordning på exemplen. Antalet möjliga kombinationer av ordningar är dock stor och det blir snabbt ohanterligt. Enbart genom att ändra på den ordning som personerna presenteras för arkitekturen ökar antalet varianter av datamängden med 6 – 24 gånger.

Resultatet mäts med samma mått som i experiment 1 (avsnitt 5.4.3), det vill säga med måtten: exakthet (Ekvation 10), känslighet (Ekvation 11), noggrannhet (Ekvation 12), falskt positiva, sant negativa, sant positiva och falskt negativa.

6 Resultat och analys

För att ge ett klart underlag för analys och höja experimentets externa validitet används två datamängder med olika egenskaper. Varje datamängd knyts till varsitt experiment. Gemensamt för de båda experimenten är att det ska utvecklas artificiella neuronnät till att klassificera en datamängd som beskriver fenomen med diffusa eller överlappande gränser i indatarymden.

6.1 Resultat för experiment 1 – syntetisk datamängd

I det första experimentet (avsnitt 5.4) används en syntetisk datamängd. Det utvecklas 100 artificiella neuronnät per arkitektur (enkelt nätverk respektive enkelt återkopplat nätverk) och datamängd (klustrad respektive ej klustrad av ARAVQ). Varje nätverk testas sedan på att klassificera tidigare osedd data och det genomsnittliga värdet (*Ekvation 13*) för varje kombination av arkitektur och datamängd räknas ut för måtten: exakthet - andelen korrekta klassifikationer (*Ekvation 10*), känslighet – andelen positiva exempel som blir klassificerade som positiva (*Ekvation 11*) och noggrannhet - andelen negativa exempel som blir klassificerade som negativa (*Ekvation 12*). Resultatet presenteras i Tabell 1.

Arkitektur	**Exakthet**	**Känslighet**	**Noggrannhet**
Enkelt nätverk utan ARAVQ	0,63	0,53	0,74
Enkelt återkopplat nätverk utan ARAVQ	0,62	0,53	0,73
Enkelt nätverk med ARAVQ	0,46	0,16	0,79
Enkelt återkopplat nätverk med ARAVQ	0,46	0,16	0,80

Tabell 1: Medelvärden (avrundade nedåt till närmaste hundradel) för resultatet vid klassificering av den syntetiska datamängden.

Den genomsnittliga exaktheten för ett enkelt nätverk utan ARAVQ är betydligt högre än för ett enkelt nätverk med ARAVQ. Resultatet är ungefär detsamma för enkelt återkopplat nätverk, med en betydligt högre exakthet utan ARAVQ än med ARAVQ.

De artificiella neuronnäten utan ARAVQ har en betydligt högre känslighet och en betydligt lägre noggrannhet än de artificiella neuronnäten med ARAVQ. Skillnaderna i känslighet är dock betydligt större än skillnaderna i noggrannhet, vilket leder till att arkitekturerna utan ARAVQ presterar ett högre antal korrekta klassifikationer. Resultatet tyder alltså inte på att det blir fler, utan snarare färre, korrekta klassificeringar för den syntetiska datamängden när den klustras av ARAVQ.

En potentiellt påverkande faktor till det låga resultatet för ARAVQ är att den är känslig för i vilken ordning exemplen presenteras, vilket i sin tur kan påverka klassificeringsresultatet. Datamängden roteras därför i 10 steg (den första tiondelen läggs sist i träningsmängden) vilket leder till 10 olika klustringar av samma datamängd. Resultatet från utvecklingen och testningen av de artificiella neuronnäten tillsammans med de olika klustringarna tyder på att ordningen inverkar på antalet korrekta klassificeringar. Det finns alltså en skillnad i exakthet mellan de olika klustringarna. Exaktheten för enkelt nätverk med ARAVQ som fått den ”bästa” klustringen är 0,48 och för enkelt återkopplat nätverk med ARAVQ 0,49. Detta är

dock en relativt liten skillnad från medelvärdet för alla kluster (Tabell 1) och den är fortfarande betydligt lägre än utan ARAVQ.

Denna skillnad i exakthet mellan de olika rotationerna av datamängden uppstår endast tillsammans med ARAVQ. När datamängderna inte förarbetas av ARAVQ finns det ingen väsentlig skillnad i exakthet.

6.2 Resultat för experiment 2 – fysiologisk datamängd

I experiment 2 utvecklas och testas artificiella neuronnät till att klassificera fysiologisk data. Den fysiologiska datamängden finns i två olika uppdelningar: en med en stor variation och en med en liten variation mellan tränings-, validerings- och exekveringsmängden (Figur 10). Med en stor variation mellan datamängderna antas det krävas en högre grad av generaliserbarhet hos den utvecklade klassificeringsarkitekturen än med en liten variation mellan datamängderna.

Det utvecklas och testas 100 artificiella neuronnät per arkitektur (enkelt nätverk respektive enkelt återkopplat nätverk), datamängd (klustrad respektive ej klustrad av ARAVQ) och version av datamängden (stor respektive liten variation), totalt 800 artificiella neuronnät. För varje kombination av arkitektur, datamängd och version av datamängden väljs det nätverk som presterat det högsta antalet korrekta klassifikationer ut till att testas på en tidigare osedd datamängd (exekveringsmängden).

6.2.1 Fysiologisk datamängd med stor variation

Den första uppdelningen av datamängden har en stor variation mellan tränings-, validerings- och exekveringsmängden. Testresultatet (Tabell 2) innehåller värden för exakthet (*Ekvation 10*), känslighet (*Ekvation 11*) samt noggrannhet (*Ekvation 12*) för det artificiella neuronnät som presterat det högsta antalet korrekta klassifikationer.

Arkitektur	Exakthet	Känslighet	Noggrannhet
Enkelt nätverk utan ARAVQ	0,75	0,41	0,76
Enkelt återkopplat nätverk utan ARAVQ	0,79	0,23	0,81
Enkelt nätverk med ARAVQ	0,30	0,77	0,29
Enkelt återkopplat nätverk med ARAVQ	0,21	0,80	0,19

Tabell 2: Resultat (avrundade nedåt till närmaste hundradel) för de bäst presterande klassificeringsarkitekturerna vid första uppdelningen av den fysiologiska datamängden. I denna uppdelning antas exekveringsmängd ha en stor variation från tränings- och valideringsmängden.

När klassificeringsarkitekturerna utvecklas och testas med den första uppdelningen av den fysiologiska datamängden är högst uppnådda exakthet för enkelt nätverk utan ARAVQ betydligt högre än för enkelt nätverk med ARAVQ. Även för enkelt återkopplat nätverket utan ARAVQ är exakthet betydligt högre än för enkelt återkopplat nätverk med ARAVQ.

Värdena för känslighet och noggrannhet tyder på att en klassificeringsarkitektur med ARAVQ har högre sannolikhet att klassificera ett positivt exempel som positivt, men lägre sannolikhet att klassificera ett negativt exempel som negativt i jämförelse med en klassificeringsarkitektur utan ARAVQ.

6.2.2 Fysiologisk datamängd med liten variation

Den andra uppdelningen av datamängden har en liten variation mellan tränings-, validerings- och exekveringsmängden. Testresultatet (Tabell 3) innehåller värden för exakthet (*Ekvation 10*), känslighet (*Ekvation 11*) samt noggrannhet (*Ekvation 12*) för det artificiella neuronnät som presterat det högsta antalet korrekta klassifikationer.

Arkitektur	Exakthet	Känslighet	Noggrannhet
Enkelt nätverk utan ARAVQ	0,76	0,82	0,42
Enkelt återkopplat nätverk utan ARAVQ	0,81	0,90	0,29
Enkelt nätverk med ARAVQ	0,67	0,69	0,52
Enkelt återkopplat nätverk med ARAVQ	0,61	0,64	0,48

Tabell 3: Resultat (avrundade nedåt till närmaste hundradel) för de bäst presterande klassificeringsarkitekturerna vid andra uppdelningen av den fysiologiska datamängden. I denna uppdelning antas exekveringsmängd ha en liten variation från tränings- och valideringsmängden.

I uppdelningen med liten variation är högst uppnådda exakthet för enkelt nätverk utan ARAVQ något högre än för enkelt nätverk med ARAVQ. Ungefär samma resultat gäller för enkelt återkopplat nätverk, där nätverket utan ARAVQ når en högre exakthet än nätverket med ARAVQ. Skillnaderna i känslighet och noggrannhet mellan teknikerna är också mindre än tidigare. Neuronnäten med ARAVQ har något högre noggrannhet medan neuronnäten utan ARAVQ har en högre känslighet.

6.2.3 Sammanfattning av experiment 2

Resultatet från experiment 2 tyder alltså på att det inte blir fler, utan snarare färre, korrekta klassificeringar när datamängden klustras av ARAVQ.

I uppdelningen med stor variation mellan tränings-, validerings- samt exekveringsmängderna presterar arkitekturerna med ARAVQ betydligt färre korrekt klassificerade exempel än i uppdelningen med liten variation mellan mängderna. För arkitekturerna utan ARAVQ är resultatet ungefär detsamma oberoende av vilken av de två uppdelningarna som används. Detta tyder på att ARAVQ har svårigheter att klustra datamängder som inte är mycket lika dem som har använts under utvecklingen. Möjligheten att generalisera lösningen till nya situationer förmodas därför vara liten.

6.3 Sammanfattning av experimentets resultat

Varken i det första eller i det andra experimentet uppnår någon av klassificeringsarkitekturerna en högre exakthet när de utvecklas och testas på datamängder som har förarbetats av ARAVQ i jämförelse med när de utvecklas och testas på datamängder som inte har förarbetats av ARAVQ. Detta tyder alltså på att ARAVQ inte framgångsrikt underlättar klassificeringsprocessen av de datamängder som används i experimentet, när den utförs med enkelt nätverk eller enkelt återkopplat nätverk.

Vidare ger datamängderna som förarbetats av ARAVQ en betydligt lägre exakthet när det finns en stor variation mellan tränings-, validerings- och exekveringsmängderna än när variationen är liten. Detta tyder på att de kluster som ARAVQ skapar inte

generaliserar till nya situationer, utan till och med motverkar generalisering som är möjlig utan ARAVQ.

7 Slutsats och diskussion

Målet med studien var att undersöka lämpligheten av att applicera ARAVQ som datareducerare för att underlätta senare klassificering med artificiella neuronnät, vilket Linåker (2003) menar att tekniken skall vara lämplig för. Undersökningen utfördes genom att testa ARAVQ på en datamängd som beskriver distinkta fenomen med diffusa eller överlappande gränser i indatarymden.

I experimentet klustrade ARAVQ datamängder som sedan användes för utveckling och testning av ett artificiellt neuronnät. Resultatet mättes i hur framgångsrikt klassificeringsarkitekturerna generaliserade sina lösningar till en tidigare osedd datamängd. Det förväntades att ARAVQ skulle reducera datamängdens omfång och lyfta fram relevanta likheter och olikheter utan att viktig information förloras, vilket underlättar utvecklingen av artificiella neuronnät som framgångsrikt klassificerar datamängden. Projektets hypotes var således att ett artificiellt neuronnät med ARAVQ som föregående steg ger lika många eller ett högre antal korrekta klassifikationer än ett artificiellt neuronnät som inte har ARAVQ som föregående steg.

Resultatet från experimentet visar dock inte på att klassificeringsarkitekturer med ARAVQ som föregående steg presterar lika många eller ett högre antal korrekta klassifikationer. Därigenom falsifieras studiens hypotes. Resultatet visar snarast på att antalet korrekta klassifikationer minskar när datamängderna förarbetats av ARAVQ. Detta är en tydlig skillnad mot resultatet i Linåker (2003), där ARAVQ framgångsrikt används i en robotikmiljö, och talar emot Linåkers spekulationer om att ARAVQ skulle kunna användas som ett generellt datautvinningsverktyg för tidsserier inom datautvinning. ARAVQ – liksom de flesta andra datautvinningsverktyg – verkar enbart passa för en viss typ av datamängd, vilket tas upp senare i diskussionen.

En bidragande orsak till det låga resultatet för ARAVQ tros vara olikheten mellan den datamängd som tekniken utvecklats med och den datamängd som den i detta projekt testats på. I Linåker (2003) används en distinkt och statisk robotikmiljö, där till exempel en korridor alltid ser ut på samma sätt (med en liten modifikation för brus). Den enkla robotikmiljön är uppbyggd av ett fåtal tydliga ”naturliga kluster” som är desamma i varje miljö. Klustren behöver därför inte generaliseras till nya situationer, utan de möter samma stimuli som de utvecklats med. Klassificering handlar däremot ofta om att hantera ”nya situationer”, eftersom det är tidigare osedda exempel som man vill göra antaganden om. Även inom analys av tidsserier hanteras ofta ”nya situationer”, om det så är fysiologisk data eller konsumentprisindex. Inom dessa områden är det alltså viktigt att lösningarna kan generaliseras till nya situationer.

I experimentet med den fysiologiska datamängden skapades två versioner av datamängden: en med en liten variation och en med stor variation mellan tränings-, validerings- och exekveringsmängderna. Versionen med en stor variation antogs kräva större möjlighet till generalisering hos klassificeringsarkitekturen än versionen med liten variation. Resultaten från experimentet tyder på att klassificeringsarkitekturerna med ARAVQ presterar betydligt sämre på data med stor variation mellan mängderna än på data med liten variation mellan mängderna. Klassificeringsarkitekturerna utan ARAVQ presterar ungefär samma resultat oberoende av variationen i datamängderna. Det här tyder alltså på att det är svårt att generalisera de kluster som skapats av ARAVQ till nya situationer, vilket i detta upplägg motverkar senare steg av bearbetningen.

Generaliseringssvårigheterna kan bero på att klustren presenteras av så kallad lokal kodning. Med lokal kodning har varje kluster lika stor avstånd till varje annat kluster vilket leder till att information om klustrens inbördes ordning förloras. När nya, tidigare osedda, beskrivningar når ARAVQ avviker de mer eller mindre från de prototyper som skapats. Är avvikelsen tillräckligt stor hamnar de utanför det korrekta klustret och kopplas istället ihop med något annat kluster. Eftersom alla kluster har lika stort avstånd till varandra försvinner all information om eventuella likheter till de andra klustren vilket minskar möjligheten för det artificiella neuronnätet att kompensera för detta. Linåker (2003) tar upp att det kan vara problematiskt med att förlora informationen, men stöter själv inte på problemet i sina experiment eftersom de består av en mycket begränsad och statisk värld som inte kräver någon större generaliseringsförmåga.

Det kan därför ifrågasättas om ARAVQ fungerar som ett generellt datautvinningsverktyg och än mer om det är lämpligt att använda vid tidsserier eller vid klassificering av ny data. Utan vidare modifiering av ARAVQ tycks den vara bäst lämpad i situationer med en statisk datamängd, där det är möjligt att utveckla arkitekturen på en datamängd som är mycket lik den som används under körningen.

Skillnaderna i uppnådda värden för noggrannhet och känslighet mellan de båda experimenten och mellan uppdelningarna i det andra experimentet är oberoende av om ARAVQ används eller ej. Detta visar snarast på egenskaper i datamängden än hos ARAVQ.

Ur ett större perspektiv visar experimentet på problematiken kring *generella* datautvinningstekniker och understryker påståendena i Pyle (1999) om att datautvinningsverktyg ska väljas beroende på vilken typ av datamängd som finns. ARAVQ har testats i de enkla miljöerna i Linåker (2003), med egenskaper som skiljer sig från de komplexa datamängderna i detta experiment. Att generalisera datareduceringstekniken, utan vidare modifikation, från de enkla miljöerna i Linåker (2003) till de komplexa datamängderna i detta experiment är antagligen att ha en övertro till tekniken.

7.1 Fortsatta studier

Problemet med generalisering kan delvis bero på att för mycket information tas bort vid klustringen, vilket innebär att det inte finns någon relation kvar mellan klustren. Dessa relationer kan dock behållas genom att lägga in ett grannskap motsvarande det som finns för ”Self-Organizing Map” (Kohonen, 1998). Med ett grannskap är det inte bara det vinnande klustret som flyttas närmare en ny beskrivning, utan även de kluster som finns i den närmaste omgivningen. Det skapas en meningsfull ordning mellan klustren, där kluster som ligger nära vararandra är lika och kluster som ligger långt ifrån varandra är olika. Att behålla relationen mellan klustren antas minska effekten av att en ny beskrivning hamnar fel och öka det artificiella neuronnätets chans att uppväga missen. Detta kan öka möjligheten till att generalisera de utvecklade lösningarna till nya situationer.

Idén med att testa ARAVQ på datamängder med beskrivningar som har diffusa eller överlappande gränser i indatarymden kan överföras till robotikdomänen. Genom att utsätta tekniken för miljöer som inte enbart består av tydligt distinkta fenomen försvåras uppgiften för ARAVQ. Till exempel kan det finnas olika typer av korridorer, där vissa smalnar av och andra blir bredare. Vidare skulle det kunna införas variation mellan beskrivningarna i den miljö som tekniken utvecklas i och beskrivningarna i den miljö som den sedan testas i. En sådan miljö skulle ha fler

likheter med de miljöer som används i Linåker (2003) än vad datamängderna i denna studie har, vilket skulle göra det möjligt att komma med fler förslag till modifikationer av ARAVQ samt mer exakt precisera på vilken typ av data tekniken är lämpig att applicera.

Avslutningsvis bör det nämnas att resultatet från ARAVQ är mycket beroende av vilka parametervärden som används. Med extrema parameterinställningar skulle antingen ett kluster skapas för varje unik indatavektor eller ett enda kluster skapas för alla indatavektorer. I det förstnämnda fallet sker ingen datareduktion alls, vilket tar bort syftet med ARAVQ. I det sistnämnda fallet försvinner all information om skillnader mellan indatavektorerna, vilket skulle hindra en kommande klassificering av indatavektorn. Med andra parameterinställningar än de som används i denna studie skulle resultatet antagligen bli annorlunda (fler eller färre korrekta klassifikationer). Antalet möjliga parameterinställningar är dock oändligt och inte möjligt att ta upp i en enda studie. För att ytterliggare belysa teknikens möjligheter och begränsningar behöver det utföras fler experiment med andra parameterinställningar och datamängder.

Referenslista

Brooks, R. A. (1991) Intelligence without reason. *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, 569-595. Sydney, Australia:

Carlsson, J. & Ziemke T. (2001) YAKS – Yet Another Khepera Simulator. I: Rückert, Sitte & Witkowski (Red.) *Autonomous Minirobots for Research and Entertainment – Proceedings of the 5th International Heinz Nixdorf Symposium,* 235-241. Paderborn, Germany: HNI – Verlagsschriftenreihe.

Defense Advanced Research Projects Agency. [Elektronisk]. Tillgänglig: http://www.darpa.mil/grandchallenge/index.htm [hämtad: 2004-04-15].

Elman, J. L. (1990) Finding Structure in Time. *Cognitive Science*, 14, 179-211.

Franklin, S. (1995). *Artificial Minds.* Cambridge, MA, USA: MIT Press.

Han, J. & Kamber, M. (2001) *Data Mining: Concepts and Techniques*, San Francisco: Morgan Kaufmann Publishers.

Hochreiter, S. & Schmidhuber, J. (1997) Long Short-Term Memory. *Neural Computation*, 9(8), 1735-1780.

Jakobi, N. (1997) Evolutionary robotics and the radical envelope of noise hypothesis. *Adaptive Behavior*, 6(2), 325-368.

Kohonen, T. (1998) The self-organizing map, *Neurocomputing*, 21, 1-6.

Linåker, F. (2003). *Unsupervised On-line Data Reduction for Memorisation and Learning in Mobile Robotics.* PhD thesis, Computer Science, University of Sheffield, UK.

The Mathworks. [Elektronisk]. Tillgänglig: http://www.mathworks.com/ [hämtad: 2004-04-05].

Mehrotra, K., Mohan, C. K. & Ranka, S. (1997). *Elements of Artificial Neural Networks.* Cambridge, MA: MIT Press.

Meeden, L. A. (1996) An incremental approach to developing intelligent neural network controllers for robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics.* 26(3), 474-485.

Mondada, R., Franzi, E. & Ienne, P. (1993) Mobile robot miniaturization: A tool for investigation in control algorithms. I: Yoshikawa & Miyazaki (Red.) *Proceedings of the Third International Symposium on Experimental Robots,* 501-513. Berlin: Springer-Verlag.

Nolfi, S. & Floreano, D. (2000) *Evolutionary Robotics - The Biology, Intelligence, and Technology of Self-Organizing Machines*. Cambridge, MA: MIT Press.

Physiological Data Modelling Contest. [Elektronisk]. Tillgänglig: http://www.cs.utexas.edu/users/sherstov/pdmc/ [hämtad: 2004-02-14].

Pyle, D. (1999) *Data Preparation for Data Mining.* San Diego, CA: Morgan Kaufmann Publishers.

Reinartz, T. (2002) A Unifying View on Instance Selection. *Data Mining and Knowledge Discovery*, 6, 191–210.

Rumelhart, D., Widrow, B. & Lehr, M. (1994) The Basic Ideas in Neural Networks. *Communications of the ACM*, 37(3), 87-92.

Ziemke, T. (2000) *Situated Neuro-Robotics and Interactive Cognition.* PhD thesis, Computer Science, University of Sheffield, UK.