FOI

Beteendemodellering med imitationsinlärning

FARZAD KAMRANI, MIKA COHEN,
FREDRIK BISSMARCK, PETER HAMMAR

FOI-R--4890--SE
ISSN 1650-1942 December 2019

Farzad Kamrani, Mika Cohen, Fredrik Bissmarck, Peter Hammar

Beteendemodellering med imitationsinlärning

Omslagsbild: Imitativt beteende hos två till tre veckor gamla spädbarn. Meltzoff A.N. & Moore M.K., Imitation of facial and manual gestures by human neonates, Science, 1977 Oct 7, med tillstånd från AAAS.

Titel	Beteendemodellering med imitationsinlärning
Title	Behaviour Modelling with Imitation Learning
Rapportnummer	FOI-R--4890--SE
Månad	December
Utgivningsår	2019
Antal sidor	34
ISSN	1650-1942
Uppdragsgivare	Försvarsmakten
Forskningsområde	Ledningsteknologi
FoT-område	Ledning och MSI
Projektnummer	E60919
Godkänd av	Cecilia Dahlgren
Ansvarig avdelning	Försvars- och säkerhetssystem
Exportkontroll	Innehållet är granskat och omfattar ingen information som är underställd exportkontrollagstiftningen

Sammanfattning

Rapporten beskriver en familj av metoder för beteendemodellering. Samtliga metoder är baserade på intuitionen att det ibland kan vara lättare att demonstrera ett önskat beteende än att formellt definiera det. Exempelvis är det enklare för en förare att demonstrera omkörning och filväxling än att manuellt bygga beteendemodeller för omkörning och filväxling. Med hjälp av dessa metoder, som går under namnet imitationsinlärning, härleds en beteendemodell automatiskt ur ett demonstrerat beteende. Utöver sedvanliga automatiseringsvinster – det förväntas kosta mindre och gå snabbare att ta fram modeller – utlovar imitationsinlärning även ett hopp om mer realistiska modeller. Mer naturligt och människolikt beteende anses vara särskilt önskvärt i militära träningssimulatorer.

Målet med arbetet är att undersöka och utvärdera om och hur imitationsinlärning kan användas för att göra datorgenererade styrkor mer realistiska och tillgängliga för Försvarsmaktens simulatorer. Rapporten utforskar ett antal olika imitationsinlärningsmetoder, däribland beteendekloning, interaktiv imitationsinlärning och inverterad förstärkt inlärning, och redogör för en serie experimentella utvärderingar av metodernas applicerbarhet i virtuella miljöer och simulatorer.

Nyckelord

Artificiell intelligens, maskininlärning, djupinlärning, simulering, datadriven beteendemodellering, datorgenererade styrkor, imitationsinlärning, beteendekloning, interaktiv imitationsinlärning, inverterad förstärkt inlärning

Abstract

This report describes a family of methods for behavioural modelling. All methods are based on the intuition that it is easier to demonstrate a desired behaviour than to formally define it. For example, it is easier for a driver to demonstrate how to switch lanes and overtake than to manually engineer behavioural models for lane switching and overtaking. Using these methods, known as imitation learning, a behavioural model is automatically inferred from demonstrated behaviour. In addition to the usual benefits that comes with automation – models can be developed more efficiently – imitation learning also holds out a promise of producing more realistic models. Models that exhibit a more natural and human-like behaviour are considered particularly desirable in military training simulators.

The aim of the work is to investigate and evaluate if and how imitation learning can be used to make computer generated forces more realistic and accessible for the Swedish Armed Forces' simulators. This report explores imitation learning methods, including behaviour cloning, interactive imitation learning and inverse reinforcement learning. The report also describes a series of experimental evaluations of these methods.

Keywords

Artificial intelligence, machine learning, deep learning, simulation, data-driven behaviour modelling, computer generated forces, imitation learning, behaviour cloning, interactive imitation learning, inverse reinforcement learning

Innehåll

1 Inledning

Datorgenererade styrkor eller agenter (termerna används utbytbart i denna rapport) som representerar mänskliga aktörer i försvarssimuleringar, spelar en allt större roll för att öka effektiviteten och minska kostnaderna i simuleringsbaserad träning, övning, planering och beslutsstöd.

Modelleringen och utvecklingen av datorgenererade styrkor är en komplex och tidskrävande process där militär domänkunskap tolkas och överförs till en programvara för att styra agenter som ersätter piloter, soldater och bemannade system. Denna styrlogik, som bestämmer vilken handling agenten ska välja bland en uppsättning av tillåtna handlingar, är agentens beteendemodell.

Beteendemodeller till datorgenererade styrkor utvecklas idag framför allt med expertdrivna metoder: skript, beteendeträd, styrlagar, m.m. Samtidigt har under senare år kompletterande, datadrivna metoder till beteendemodellering väckt stor uppmärksamhet med bl.a. en rad resultat i de två mest ansedda tidskrifterna Nature [1, 2, 3, 4] och Science [5] och en mängd artiklar i populärmedia. Istället för att en beteendemodell programmeras med stöd av en expert härleds beteendemodellen automatiskt ur data från en demonstration av beteendet. Intuitionen är att det är lättare att demonstrera ett beteende än att formellt definiera det.

Datadriven beteendemodellering kan ses som en form av automatiserad beteendemodellering. Utöver sedvanliga automatiseringsvinster – det förväntas kosta mindre och gå snabbare att ta fram modeller – inger datadriven beteendemodellering även hopp om mer realistiska modeller, inte minst modeller som uppvisar ett mer naturligt, människolikt beteende, något som är särskilt önskvärt i just militära träningssimulatorer.

Denna rapport presenterar en serie fallstudier som genomförts med olika former av datadriven beteendemodellering utifrån demonstrationsdata, s.k. imitationsinlärning. Flertalet fallstudier genomfördes i *Virtual Battlespace 3* (VBS3), en träningssimulator som används inom Försvarsmakten. Metoderna är dock inte specifika för VBS3 utan kan överföras till andra simulatorer; några av metoderna har sedermera även använts i andra virtuella simulatorer med mindre anpassningar i den underliggande koden [6]. Metoderna är inte heller specifika för de två beteenden som fallstudierna syftar till – att styra ett fordon på ett mänskligt sätt på en godtycklig vägbana samt att ruttplanera med hänsyn till intuitiva, icke-formaliserade taktiska prioriteringar; några av metoderna har senare använts för att skapa en pilotbeteendemodell hos Flygvapnets luftstridssimuleringscentrum (FLSC) [7].

1.1 Syfte och mål

Denna rapport syftar till att undersöka möjligheten att modellera datorgenerade styrkor med imitationsinlärning. Det traditionella angreppssättet att

modellera agenter är en komplex, tidskrävande och kostsam uppgift då de regler som utgör beteendemodellen måste skapas för hand av programmerare med stöd av domänkunskap som kan vara svårt att definiera.

Målet med detta arbete är att som ett alternativ till det traditionella angreppssättet, forska om och utvärdera olika imitationsinlärningsmetoder och förfina dessa metoder för att skapa agenter och testa dessa i några för Försvarsmakten relevanta sammanhang och simulatorer.

1.2 Rapportens målgrupp

Rapporten är skriven för att kunna läsas och förstås i sin helhet utan några förkunskaper inom imitationsinlärning. Alla med ett allmänt intresse för AI och dess tillämpningar, speciellt modellering och simulering, kan ha nytta av att läsa rapporten.

1.3 Metod

Vi har studerat ett antal uppmärksammade metoder inom imitationsinlärning och applicerat dem på för Försvarsmakten relevanta frågor och simuleringssituationer. Vidare har vi undersökt möjligheten att modifiera och utveckla metoderna så att de är mer anpassade till tillämpningarnas förutsättningar vad gäller tillgänglighet till data och beräkningskraft.

Rapporten ämnar inte ge en heltäckande bild av de presenterade metoderna, utan snarare tillhandahålla en övergripande redogörelse för dem. Den intresserade hänvisas till rapporterna för de examensarbeten som har utförts inom ramen för projektet och även de vetenskapliga artiklar som refereras.

1.4 Rapportens upplägg

I nästa kapitel förklaras de prövade metoderna och teknikerna, alltså beteendekloning, interaktiv imitationsinlärning och de metoder vi testat som en utökning av imitationsinlärning, samt inverterad förstärkt inlärning. Detta följs av ett kapitel som presenterar de experiment som utförts. Rapporten avslutas med diskussion och slutsatser.

2 Imitationsinlärning

Vi studerar beteendemodellering genom att lära upp en agent (även kallad *målagent*) utifrån beteendet hos en annan agent (oftast kallad *expertagent*). Det kan finnas flera skäl att föredra detta sätt att modellera beteenden, t.ex. kan det vara svårt att specificera ett beteende men relativt sett enklare att inhämta data från en mänsklig agent som utför samma uppgift, t.ex. kör en bil. En annan anledning kan vara att målet är att skapa en modell som beteendemässigt är så lik en annan (mänsklig) agent som möjligt.

Vi är intresserade av beteenden som manifesterar sig som en sekvens av handlingar som påverkar agentens omgivning eller agenten själv. Inlärning innebär då att träna en modell av en experts beteende utifrån en följd av handlingar och resulterande påverkan. Denna typ av maskininlärning kallas imitationsinlärning. Detta är ganska likt den mer kända maskininlärningsmetoden *övervakad inlärning* (eng. *supervised learning*), där en modell tränas att känna igen ett tillstånd och producera ett svar givet ett känt facit. Men till skillnad från övervakad inlärning görs detta sekventiellt och agenten fattar följder av beslut i en omgivning där besluten påverkar systemets framtida tillstånd [8].

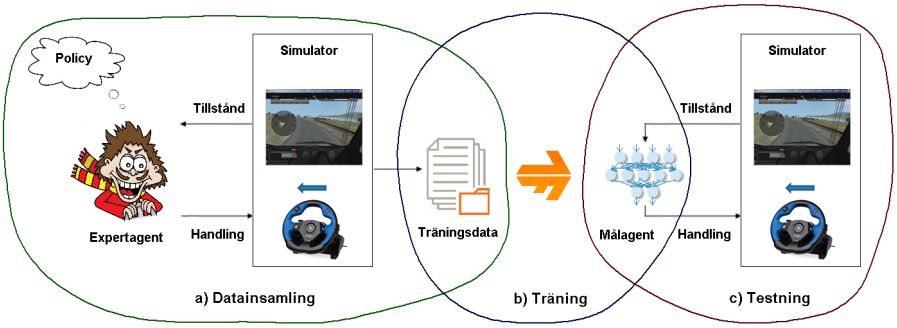
För tydlighetens skull förklarar vi följande begrepp som används i imitationsinlärning:

- *Expertagent* eller *referensagent* (förkortas även till *expert* respektive *referens*) är den agent vars beteende ska imiteras. Den är oftast en mänsklig demonstratör men i det allmänna fallet kan experten vara vilken typ av agent som helst.
- *Målagent* (eller *novisagent*, eller kort och gott *agent* om det framgår av sammanhanget vilken typ av agent som avses) är den agent som imiterar experten.
- *Omgivning* är den miljö som expert- och målagenten interagerar med. Omgivningen kan påverkas av agentens handlingar och agenten kan i sin tur påverkas av omgivningen. Eftersom vårt fokus är simuleringsagenter består omgivningen av den virtuella miljö som finns representerad i simulatorn.
- *Tillstånd* är en agents interna representation, dvs. den information som agenten använder för att välja sin nästa handling. I datorspel utgörs tillstånden ofta av ögonblicksbilder av det pågående spelet med information om agentens egen position, dess hälsotillstånd, andra agenters positioner, m.m. Tillståndsrummet är mängden av alla tillstånd som agenten kan hamna i.
- *Handling* är det agerande som en agent kan välja att utföra i varje tillstånd. Handlingsrummet är mängden av alla tillgängliga handlingar.
- *Träningsdata* är data som är samlad i förväg genom att låta experten interagera med miljön under en förutbestämd tid, dvs. en serie av tillstånd

och handlingar (det första tillståndet samt handlingen som utfördes där, följt av det resulterande andra tillståndet och handlingen som utfördes där, etc.). I vissa fall kan ny träningsdata samlas även under pågående träning, vilket kräver att experten även är tillgänglig under träningen.

- *Policy* är en teknisk benämning på det beteende som avses fångas, en uppsättning regler som mappar tillståndsrummet till handlingsrummet. För en agent i ett datorspel motsvarar agentens policy de signaler som styr agenten i spelet, ekvivalenta med de kommandon en mänsklig spelare skulle ge en agent i spelet i samma situation.

För att belysa imitationsinlärningsmetoder kommer vi i detta kapitel att genomgående använda oss av ett exempel med en agent som tränas att styra ett fordon i en simulator. Expertagenten är en mänsklig förare som styr fordonet i simulatorn. Vi väljer ett *end-to-end* tillvägagångssätt (om inte annat explicit anges), vilket betyder att vi använder skärmbilder av förarvyn som agentens tillstånd istället för parametrar som beskriver fordonets placering på vägbanan eller andra värden. Dessa bilder sparas tillsammans med respektive utslag på styrspaken som använts för att styra fordonet i rätt riktning i varje givet tillstånd. För enkelhets skull antar vi att fordonet alltid körs med en konstant hastighet och agentens handling endast är styrspakens utslag. Träningsdata är uppsättningen av samlade bilder och styrspaksdata. En policy är i detta fall således en funktion som anger hur mycket styrspaken skall vridas för varje möjlig förarvy. Träningsdata är en ändlig uppsättning av fordonets tillstånd (givet som skärmbilder) som har sparats tillsammans med respektive värde på styrspakens utslag, och som producerats genom att en expertagent styrt fordonet under ett bestämt tidsintervall (figur 2.1).



Figur 2.1 – Olika steg inom imitationsinlärning: a) datainsamling genom att låta en expert köra ett fordon i simulatorn, b) träning av målagenten med hjälp av de insamlade data, och c) testning av målagenten genom att låta den köra samma fordon i simulatorn.

I detta kapitel beskriver vi imitationsinlärningsmetoder som har presenterats i den vetenskapliga litteraturen samt potentiella förbättringar av dessa. I

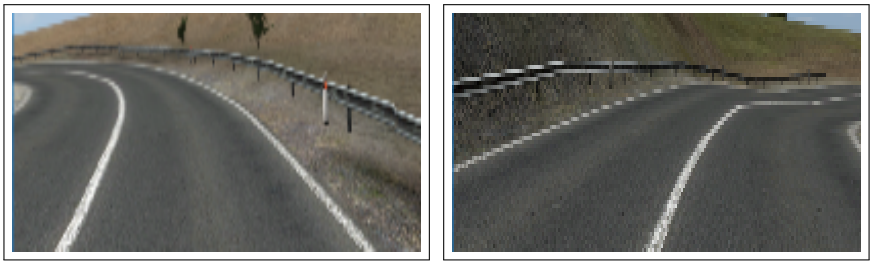
de kommande avsnitten ligger vårt fokus på förklaring av metoderna.

2.1 Beteendekloning

Den enklaste formen av imitationsinlärning, beteendekloning, är en reducering av inlärningsproblemet till övervakad inlärning. Reduktionen bygger på ett förenklat antagande om att data är *oberoende och likafördelad* (eng. *independent and identically distributed*), vilket innebär dels att nästkommande tillstånd antas vara oberoende av modellens prediktion och dels att olika slags tillstånd antas komma med samma frekvenser under träningen som vid testning och användning [9].

Dessa antaganden är problematiska för imitationsinlärning. När en agent interagerar med en omgivning påverkar agentens handling (beteendemodellens prediktion) det efterföljande tillståndet, vilket i sin tur medför att målagenten kan komma att driva systemet till helt andra delar av tillståndsrummet vid testning och användning än expertagenten under träningen. I sedvanlig övervakad inlärning, säg klassificering av bilder, påverkar inte en felaktig prediktion de kommande prediktionerna, men i imitationsinlärning medför en felaktig prediktion att agenten hamnar i tillstånd som inte har observerats förut, vilket ökar risken något för ytterligare felaktiga prediktioner, vilket i sin tur ökar risken än mer för ytterligare felaktiga prediktioner, osv. – vi får ett självförstärkande fel [10]. I värsta fall kan agenten hamna i ett oönskat tillstånd som den inte kan ta sig ur.

Figur 2.2 visar skillnaden mellan ett normalt tillstånd (2.2a) där fordonet befinner sig i en situation som agenten kan hantera och ett kritiskt tillstånd (2.2b) som skiljer sig alltför mycket från träningsdata. Eftersom agenten inte har tränats på det kritiska tillståndet finns det stor sannolikhet att handlingar utifrån det inlärda beteendet inte styr tillbaka, utan leder till att fordonet hamnar utanför körbanan.



(a) Normalt tillstånd (b) Kritiskt tillstånd

Figur 2.2 – Två olika typer av tillstånd (a) ett normalt tillstånd, och (b) ett kritiskt tillstånd där fordonet är på väg in i motsatt körbana.

Detta gäller naturligtvis inte om tillståndsrummet är begränsat och ex-

perten besöker hela tillståndsrummet under datainsamlingen. Om så är fallet, hamnar agenten alltid i ett tidigare observerat tillstånd även om en felaktig handling tar agenten till ett oönskat tillstånd (på väg utanför körbanan). I sådana fall kan man använda beteendekloning utan någon risk.

Beteendekloning är enkel och effektiv och kan användas i begränsade situationer, speciellt när tillstånd som agenten kommer stöta på inte avviker nämnvärt från träningsdata.

2.2 Interaktiv imitationsinlärning

För att överkomma problemet med det självförstärkande felet som observeras i beteendekloning har flera metoder föreslagits, bland dem är DAgger-algoritmen [11] en av de mest diskuterade och uppmärksammade. Namnet DAgger kommer från *Dataset Aggregation* som är en iterativ algoritm där en expertagent och en novisagent är engagerade samtidigt. Novisagenten initialiseras slumpmässigt eller tränas med något initialt träningsdata. Efter denna initialisering, växlar DAgger mellan att iterativt samla in ytterligare träningsdata från en blandning av expert- och novisagenten och att uppdatera sin policy utifrån den totala mängden insamlad data. Under en given iteration interagerar en agent med omgivningen (här simulatorn) under övervakning av en beslutsregel som i varje tillfälle där agenten behöver välja en handling, bestämmer om expert- eller novisagentens handling ska användas för att interagera med omgivningen. De observerade tillstånden (i fallet självkörande fordon, skärmbilder) som samlas in under varje iteration vid de tidpunkter då experten styrde, och expertens motsvarande handlingar utgör en ny datauppsättning som kombineras med de föregående uppsättningarna och används för att träna om modellen (målagenten) i slutet av varje iteration. Genom att låta novisagenten styra under träningen kommer den att visa vilka tillstånd dess beteende tar agenten till och ger då experten möjlighet att korrigera eller visa på ett bättre beteende om det behövs. Itereringen fortsätter tills modellen är tillräcklig bra eller tid och beräkningsresurser sätter en gräns [12].

I den ursprungliga DAgger-algoritmen [10] (oftast kallad VanillaDAgger) är beslutsregeln som avgör vilken modell som ska interagera med omgivningen ett slumptal som dras i varje beslutsögonblick och jämförs med ett tröskelvärde så att novisagentens handling används om slumptalet är större än tröskelvärdet. Vid start sätts tröskelvärdet till ett högt tal, vilket innebär att experten oftast styr fordonet. Värdet minskas successivt i varje iteration och leder till att kontrollen överlämnas mer till novisagenten med tiden och expertens inblandning minskas.

Det som skiljer de olika variationerna av DAgger-algoritmen är beslutsregeln som avgör under vilka omständigheter experten eller novisagenten får ta kontrollen och interagera med omgivningen. VanillaDAggers beslutsregel, en avtagande sannolikhet, är enkel men har den nackdelen att den inte tar hänsyn till "kvaliteten" av agentens beslut. Även om novisagenten föreslår en olämplig handling (t.ex. styr fordonet på ett farligt sätt) finns det viss sannolikhet

att VanillaDAgger använder novisagentens handling. Dessutom kräver VanillaDAgger att en expertagent finns med och utnyttjas under hela träningen.

Zhang och Cho [13] föreslår en utvidgning av DAgger, kallad *SafeDAgger*, som är en mer effektiv metod (i termer av behov av expert) för end-to-end träning av självkörande agenter. Den viktigaste skillnaden mellan SafeDAgger och VanillaDAgger är att i SafeDAgger tränas en *testfunktion* som används istället för den slumpmässiga beslutsregeln i VanillaDAgger för att bestämma vilken agent ska ha kontroll över systemet. Vid varje tidpunkt avgör testfunktionen om det är säkert att låta novisagenten köra och om så inte är fallet låter man experten köra istället. Det är alltså testfunktionen som avgör om man kan lita på det val som görs av novisagenten i det aktuella tillståndet. Det är viktigt att poängtera att detta beslut tas utan att fråga experten. Fördelarna kommer av att det är mycket lättare att träna en testfunktion (som utgör en beslutsgräns mellan säkra och osäkra tillstånd) än att träna en komplex mappning från ett tillstånd till en styrvariabel, dvs. det är lättare att lära in en bedömning om en situation är säker eller osäker än att lära sig vilken handling är optimal för situationen. I inlärning av självkörande fordon kan testfunktionen exempelvis lära sig att skilja mellan en ödslig väg och en livligt trafikerad väg och bestämma att det är säkert att låta novisagenten köra på den ödsliga vägen, men inte på den livligt trafikerade vägen.

Ett problem med SafeDAgger är att testfunktionen som används i beslutsregeln oftast är baserad på avvikelse från expertpolicyn. Detta innebär att novisagenten tvingas att hålla sig nära expertens agerande oavsett om den befinner sig i ett säkert eller riskfyllt tillstånd, vilket medför att agentens möjligheter att utforska olika handlingar antingen hålls mycket begränsat även om det inte föreligger någon risk eller att handlingsutrymmet blir för stort när det faktiskt är riskfyllt.

En annan variant på utvidgning av DAgger-algoritmen som försöker komma till rätta med detta kallas *EnsembleDAgger* [12]. Idén i denna metod är att utöver att jämföra skillnaden mellan experten och novisagentens handlingar, också titta på osäkerheten i novisagentens handling i varje tillstånd. Metoden går ut på att låta den upplärda agenten styras av en mängd "delagenter", där varje delagent tränas på samma sätt men med delvis olika träningsdata och agentens beteende väljs som medelvärdet av delagenternas beteende. En modell för osäkerhet i agentens handling skapas genom att beräkna variansen för delagenternas handlingar. Avvikelsen från expertpolicyn och osäkerheten måste båda vara mindre än två valda gränsvärden för att novisagentens handling skall väljas i respektive steg i träningen, i annat fall bedöms det riskfyllt och experten får styra istället.

2.3 Belöningsorienterad imitationsinlärning

Det finns problem med olika versioner av DAgger-algoritmen. Medan VanillaDAgger är för likgiltig och låter novisagenten styra fordonet trots att det kan leda till oönskade situationer, kan SafeDAgger bli för konservativ, vilket leder

till att novisagenten inte utforskar tillräckligt mycket och inte tränas på hur den skall agera i oönskade tillstånd, t.ex. riskfyllda situationer då fordonet håller på att köra av vägen. Båda dessa problem leder till att algoritmerna kräver mycket träningsdata från en expert. I studerade exempel behöver Ensemble-DAgger förvisso mindre träningsdata från experten än de andra varianterna av DAgger, men lider i övrigt av liknande problem. Osäkerhetsmåttet som används för att hindra novisagenten att utföra olämpliga handlingar begränsar den till att ligga nära expertens förevisade handlingar och begränsar därför hur mycket novisagenten får möjlighet att se olika tillstånd. Detta leder till frågan om det går att hitta en bättre beslutsregel för DAgger-algoritmen.

Vi föreslår en modifiering av DAgger-algoritmen genom att införa en annan konstruktionen av beslutsregeln. Målet är att låta novisagenten pröva möjliga handlingar och samtidigt begränsa risken för att hamna i olämpliga tillstånd. Detta åstadkommes genom att införa en testfunktion på motsvarande sätt som i SafeDAgger, med skillnaden att denna inte ges av träningsdata för hur experten agerar, utan definieras separat. Då SafeDAgger lär vad som är "säkert" utifrån det beteende som experten förevisat i träningsdata är idén här att låta en modell guida agenten så att den utforskar tillståndsrummet utan att hamna i alltför riskfyllda tillstånd. Modellen som guidar agenten är en funktion som mappar tillståndsrummet till en belöning (egentligen en kostnad som är den motsatta till belöning), vars värde minskar när fordonet hamnar i riskfyllda tillstånd.

Det är tänkbart att med maskininlärning träna upp en modell för riskfyllda situationer, exempelvis genom att låta en expert klassificera en mängd tillstånd (jfr. figur 2.2), och sedan använda denna modell i beslutsregeln. I praktiken valdes dock att manuellt bygga in domänkunskap i beslutsregeln. I inlärning av självkörande fordon innebär detta att riktningen av fordonet och dess närhet till vägkanten inkluderas i designen av beslutsregeln. Vi kallar denna algoritm för belöningsorienterad imitationsinlärning förkortad till *RG-DAgger (Reward-Guided DAgger)*, eftersom den inkorporerar en belöning i beslutsregeln.

Under inlärningen tränas agenten genom att styra fordonet på en väg, och en modell av fordonets tillstånd på vägen bedömer om fordonet är i ett säkert eller osäkert tillstånd. Modellen använder fordonets riktning på vägen och avståndet till vägens kant som mått för att definiera säker körning. Om tillståndet bedöms att inte vara omedelbart riskabel tillåts novisagenten att styra fordonet och på så sätt utforska tillståndsrummet. På motsvarande sätt som ett slumptal används i VanillaDAgger för att avgöra om novisagenten skall få fortsätta styra eller om experten skall ta över, påverkar en belöning beslutet i RG-DAgger.

Denna beslutsregel medför att ju närmare vägens kant fordonet befinner sig, desto större sannolikhet att låta experten köra, och ju större skillnaden är mellan vägens riktning och fordonets riktning, desto större sannolikhet att experten får ta över.

Även om data från simuleringen (vägens kanter och vägens riktning) nyttjas för att beräkna beslutsgränsen, används, precis som för SafeDAgger, dessa extra data endast för att beräkna beslutsgränsen och inlärningen av beteendet sker

endast med information från förarens vy.

Medan VanillaDAgger behöver expertens styrning under hela träningen (oavsett om det är expert- eller novisagenten som kontrollerar fordonet), behövs experten i RG-DAgger endast vara tillgänglig för att styra fordonet vid de punkter där beslutsregeln ger den kontrollen.

En mer detaljerad redogörelse för belöningsorienterad imitationsinlärning återfinns i examensarbetesrapporten [14] som tagits fram inom ramen för detta arbete.

2.4 Stegvis imitationsinlärning

DAgger försöker att lösa problemet med diskrepansen mellan fördelningen av träningsdata och testdata genom att iterativt samla in ny data och träna om målagenten på aggregerade data. Detta görs eftersom om träningsdata och testdata inte har samma fördelning, riskerar agenten att stöta på tillstånd som inte finns representerade i träningsdata. Ett problem med samtliga varianter av DAgger är att aggregeringen av data leder till ökande träningstider och ökande minneskrav. Dessutom kräver DAgger att träningsdata ska bevaras för all framtid om modellen behöver tränas för att hantera en ny situation. Anledningen till att modellen måste tränas om på den aggregerade datauppsättningen istället för enbart på ny data är att man vill undvika något som kallas för *katastrofal glömska* [15, 16] (eng. *catastrophic forgetting*).

Alla metoder för imitationsinlärning som beskrivs i denna rapport använder maskininlärningsmetoder som bygger på djupa artificiella neuronnät, och katastrofal glömska är ett välkänt problem för neuronnätbaserade modeller i allmänhet och är inte specifikt för imitationsinlärning. Inlärningsmetoder som använder sig av neuronnät lider av ett inneboende problem. Nätverk som tränas sekventiellt på två uppgifter, även om uppgifterna liknar varandra, har en tendens att helt glömma tidigare inlärda uppgifter. Effekten av detta blir att ett tidigare inlärt beteende helt kan försvinna redan efter tilläggsträning på en relativt liten mängd ny data.

Det har föreslagits flera metoder för att lösa eller förmildra katastrofal glömska så att neuronnät kan lära sig nya uppgifter utan att glömma tidigare inlärda uppgifter. Dessa metoder går under benämningen *stegvis inlärning* (eng. *continual learning*).

En metod för stegvis inlärning som har föreslagits av forskarna på DeepMind är *elastisk viktkonsolidering*, eng. *elastic weight consolidation* (EWC) [17]. Metoden är inspirerad av teorier från neurologi, specifikt idén att hjärnan skulle fungera så att neuroplasticiteten hos de nervceller som varit viktiga för tidigare inlärda uppgifter minskas vid ny inlärning. Algoritmen är implementerad så att redan inlärda viktparametrar i neuronnätet är begränsade vid ny inlärning och deras värde håller sig nära sina gamla värden. Detta är principiellt möjligt eftersom neuronnät i allmänhet har kapaciteten att lagra mycket mer information än vad som behövs. Vid ny inlärning separerar algoritmen viktiga från oviktiga neuroner och uppdaterar endast de som är oviktiga för den gamla

uppgiften.

Vi har undersökt möjligheten att förbättra SafeDAgger-algoritmen (se avsnitt 2.2) med stegvis inlärning för att skapa en mer skalbar och flexibel algoritm. Idén går ut på att istället för att aggregera ny data med befintlig data använda en stegvis inlärningsalgoritm, närmare bestämt elastisk viktkonsolidering, för att skydda modellens tidigare inlärda kunskaper och därigenom enbart träna den på nya data. Metoden gör det möjligt att ha kortare träningstider och mindre minneskrav. Dessutom behöver man inte lagra gamla data för evigt. Dessa egenskaper gör att den föreslagna metoden är fördelaktig jämfört med den ursprungliga SafeDAgger-algoritmen. Vi kallar metoden som är en kombination av EWC (elastisk viktkonsolidering) och SafeDAgger för EWC-SD.

Stegvis imitationsinlärning finns vidare beskriven i en examensarbetesrapport [18] som tagits fram inom detta arbete.

2.5 Inverterad förstärkt inlärning

Imitationsinlärningsteknikerna som diskuterats hittills – beteendekloning och varianter av DAgger – kan ha svårt att lära sig ett mer långsiktigt, målinriktad beteende. *Inverterad förstärkt inlärning* [19] är en form av imitationsinlärning som försöker att adressera detta.

Med inverterad förstärkt inlärning härleds det bakomliggande målet som driver ett demonstrerat beteende, varför metoden kan ses som en slags invertering av *förstärkt inlärning* – därav namnet. Med förstärkt inlärning genereras ett beteende som uppfyller ett givet mål, mer specifikt ett beteende som långsiktigt optimerar en given belöningsfunktion (alternativt minimerar en given kostnadsfunktion) [20].

När det bakomliggande målet är härlett med inverterad förstärkt inlärning kan i ett nästa steg ett beteende genereras med förstärkt inlärning eller annan optimering. Metoden kan vid en första anblick förefalla konstlad och omständlig, men ofta är det belöningsfunktionen, snarare än policyn, som är den mest koncisa, robusta och överförbara definitionen av en uppgift [19]. Belöningsfunktionen kan även skapa en förståelse för hur experten beter sig [21].

I enkla fall då det underliggande målet är välkänt och lätt att specificera är det överflödigt att härleda målet ur demonstrerat beteende. Men i andra fall kan det vara svårt att i detalj precisera målet för ett beteende som ändå är lätt att demonstrera. Människor kan t.ex. ruttplanera utifrån det långsiktiga målet att minimera en viss implicit, inte fullt verbaliserad kostnad som kan vara svår att specificera. För t.ex. en militärtransport kanske det gäller att beakta visibilitet, skydd, hinder, m.m. [22], men hur dessa faktorer ska estimeras utifrån kartan eller hur de ska viktas och vägas samman kan vara svårdefinierat; den erfarna militären löser detta med magkänsla.

Inverterad förstärkt inlärning antar att det observerade beteendet är rationellt: beteendet som observerats antas minimera en viss okänd kostnadsfunktion. Med detta antagande reduceras imitationsinlärningsproblemet (t.ex. problemet att efterlikna en människas ruttplanering) till problemet att estimera

den okända kostnadsfunktionen utifrån det observerade beteendet (t.ex. utifrån observerade rutter). Hur estimeringen går till varierar mellan olika former av inverterad förstärkt inlärning. Flera av de mest uppmärksammade metoderna estimerar kostnadsfunktionen med hjälp av en generalisering [23] av den statistiska principen om maximal entropi [24].

3 Tillämpningar i simulatorer

I detta kapitel beskrivs de experiment och fallstudier som vi har utfört för att få en bättre förståelse och testa tillämpbarheten av både de ursprungliga metoderna och de modifieringar som vi föreslår inom områden beteendekloning (avsnitt 2.1), interaktiv imitationsinlärning (avsnitt 2.2), belöningsorienterad imitationsinlärning (avsnitt 2.3) och stegvis imitationsinlärning (avsnitt 2.4). Även experiment kopplat till inverterad förstärkt inlärning, (avsnitt 2.5), återges. Var och ett av de följande fem avsnitten behandlar experiment utfört med metoderna i respektive avsnitt ovan. Samtliga experiment utom det sista berör träning av en syntetisk förare att imitera ett inspelat mänskligt förarbeteende i träningssimulatorn VBS3. Det sista experimentet handlar om försök med ruttplanering utifrån metoden med inverterad förstärkt inlärning.

3.1 Experiment med beteendekloning

Beteendekloning är den enklaste av de presenterade teknikerna för imitationsinlärning. Vi har utfört en serie försök med beteendekloning. Dessa har tidigare presenterats i [25, 26], men återges i detta avsnitt som en grund mot vilken de andra metoderna kan jämföras. I dessa försök tränades en agent att imitera ett inspelat mänskligt förarbeteende i VBS3. Träningsdata anskaffades genom att en mänsklig spelare framförde fordonet i VBS3 i 20 minuter samtidigt som skärmbilder (förarvyn) kontinuerligt sparades och indata från den styrspak som styrde fordonet registrerades. Dessa inspelade data användes för att träna ett djupt neuronnät som sedan visade sig vara kapabelt att styra fordonet i simulatorn, hantera högertrafik (figur 3.1a), köra i omgivningar den ej tränats på (figur 3.1b), samt styra fordonet mot vägen och på rätt bana efter avsiktligt injicerade avkörningar under enkla förutsättningar (figur 3.1c).

(a) Högertrafik (b) Generalisering på ny bana (c) Styrning mot vägen efter avkörning i enkel miljö

Figur 3.1 – Ett exempel där beteendekloning tillämpats för att imitera mänskligt förarbeteende i träningssimulatorn VBS3.

Beteendekloning tillämpades *end-to-end* vilket i detta sammanhang innebär att modellen tränades med endast skärmbilder och de tillhörande mänskliga styrningen som indata. Inlärningsprocessen behövde inte analysera bilderna och extrahera olika detaljer eller känna till de interna representationerna av olika steg inom simuleringen. Modellen lärde sig helt enkelt hur mycket den ska styra till vänster eller höger given den aktuella förarvyn (skärmbildens pixeldata).

I detta experiment beter sig den tränade modellen på ett realistiskt sätt och man kan känna igen det körsätt som finns i inspelad data. Dock kan modellen ha svårt att generalisera till situationer som skiljer sig alltför mycket från inspelad data och misslyckas ibland att hålla fordonet på banan i en annan omgivning där detaljerna är för annorlunda.

Detta är inte någon slump utan beror på att vi har hanterat problemet som en övervakad inlärning och antagit att träningsdata och testdata är oberoende och likafördelade, vilket inte är korrekt och kan leda till ett självförstärkande fel. Med andra ord, trots att modellen åtnjuter en relativ god generaliseringsförmåga (producerar rätt svar även på data som inte tidigare har tränats på) kan den köra av den nya banan under vissa förhållanden.

Experimenten bekräftar de styrkor och begränsningar som är karakteristiska för beteendekloning. Det är relativt enkelt och det krävs inte mycket data för att träna modellen till att i begränsade situationer kunna kontrollera fordonet, samtidigt som ett självförstärkande fel kan leda till att fordonet hamnar i katastrofala tillstånd och riskerar köra av vägen.

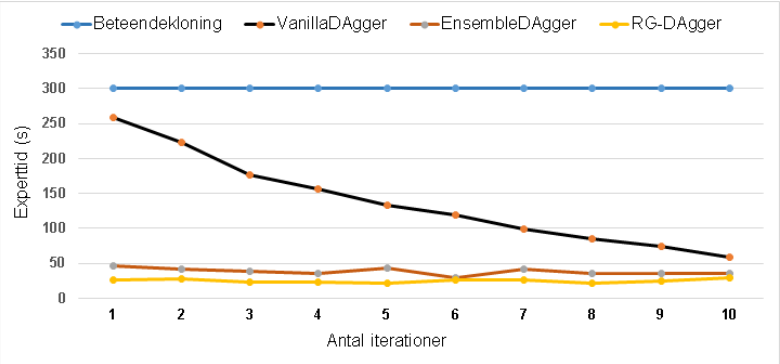
3.2 Experiment med interaktiv imitationsinlärning

I detta avsnitt redovisar vi resultat av experiment med interaktiv imitationsinlärningsmetoder som beskrevs i avsnitt 2.2. Vi har inte gjort en uttömmande utvärdering av olika DAgger varianter. Av flera skäl är det relativt svårt att utvärdera DAgger-algoritmen och jämföra dess olika varianter. Bland annat för att det inte är klart hur man definierar ett mått på prestanda som kan användas för att utvärdera likheten mellan expert- och novisagentens beteende även om man reducerar problemet till det till synes enkla fallet om likheten mellan två fysiska banor.

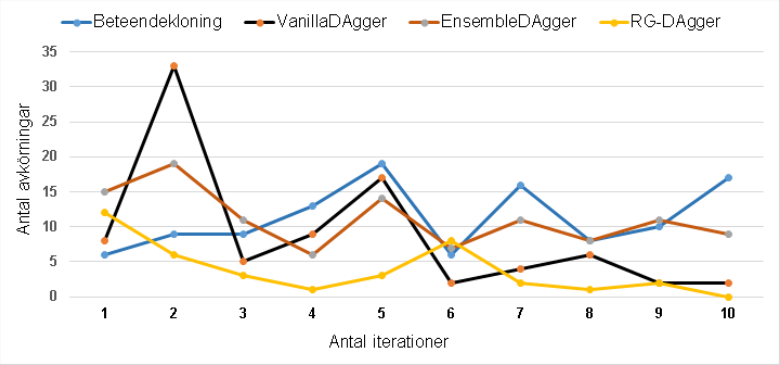
Vi har dock jämfört beteendekloning (som en baseline), VanillaDAgger och EnsembleDAgger med avseende på dels experttid, dvs. den tid som expertens handlingar användes för att styra fordonet i simuleringen och dels antal avkörningar som sker på testbanan i varje iteration.

Testbanan är en 1,67 km lång bana som inte har setts tidigare av agenten och fordonet körs med en konstant hastighet på 20 km/t, vilket innebär att simuleringstiden blir 300 sekunder. Vid en avkörning placeras fordonet på vägbanan igen och dess riktning rättas till så att den ska kunna fortsätta sin färd. Resultaten har sammanställts i figur 3.2 (även resultat för RG-DAgger som diskuteras i nästa avsnitt återfinns i denna figur).

Som figur 3.2a visar kräver beteendekloning att experten styr fordonet hela



(a) Experttid som funktion av antalet iterationer vid träning



(b) Antalet avkörningar under testning som funktion av antalet iterationer vid träning

Figur 3.2 – Experttid i sekunder vid träning (a), och antalet avkörningar under testning (b), för modeller som tränas från 1 till 10 iterationer med beteendekloning, VanillaDAgger, EnsembleDAgger och RG-DAgger.

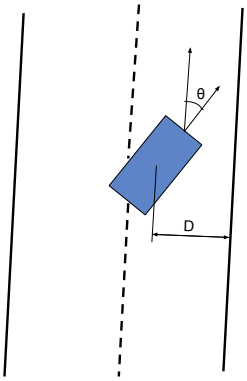
tiden, dvs. 300 sekunder. I VanillaDAgger minskas detta värde ständigt, men detta kan inte betraktas som ett mått på algoritmens prestanda utan beror bara på att det tröskelvärde som bestämmer när experten ska styra fordonet minskas kontinuerligt enligt algoritmen. Experttiden för EnsembleDAgger börjar med ett lågt värde, ca 46 sekunder och minskar sakta men förhållandevis stadigt till ca 36 sekunder efter 10 iterationer.

I figur 3.2b illustreras antalet avkörningar på samma testbana. VanillaDAgger har en relativt bättre prestanda jämfört med EnsembleDAgger som har nästan lika många avkörningar som beteendekloning. Med andra ord visar både VanillaDAgger och EnsembleDAgger en bättre prestanda än beteendekloning, men jämförelsen mellan dessa två DAgger-algoritmer emellan ger inte ett entydigt resultat. Medan VanillaDAgger visar bättre resultat än EnsembleDAg-

ger på antalet avkörningar är förhållandet det motsatta när det gäller behovet av experttid.

3.3 Experiment med belöningsorienterad imitationsinlärning

Belöningsorienterad imitationsinlärning (RG-DAgger) (avsnitt 2.3) är en metod som vi föreslår som en modifiering av DAgger-algoritmen där fordonets placering och riktning påverkar beslutsregeln som avgör om novisagenten eller experten ska styra fordonet. En större vinkel mellan fordonet och vägens riktning betyder en större sannolikhet att kontrollen överlämnas till experten. På samma sätt ökar denna sannolikhet med ett minskat avstånd mellan fordonet och vägkanten. Figur 3.3 tydliggör hur fordonets placering och riktning mäts.



Figur 3.3 – Vinkeln mellan fordonet och vägens riktning, θ, och avståndet till vägkanten, D, påverkar sannolikheten att tillståndet bedöms osäkert och styrningen överlämnas till experten.

I ett experiment med samma uppsättning som tidigare jämförs RG-DAgger med beteendekloning, VanillaDAgger och EnsembleDAgger på dels experttid, dvs. den tid som expertens handlingar användes för att styra fordonet i simuleringen och dels antal avkörningar i varje iteration som sker på en 1,67 km testbana som inte har setts under träningen.

Resultaten har sammanställts i figur 3.2, dvs. figuren som användes för andra DAgger-algoritmer (avsnitt 3.2). Som figur 3.2a visar kräver RG-DAgger mindre experttid än beteendekloning, VanillaDAgger och EnsembleDAgger. Det börjar med ett lågt värde dvs. ca 26 sekunder redan i första iterationen och trots små variationer ligger det under 30 sekunder i alla 10 iterationer med det minsta värdet på ca 21 sekunder i åttonde iterationen. Figur 3.2b illustrerar antalet avkörningar på samma testbana. RG-DAgger visar minst antal avåkningar följt av VanillaDAgger medan EnsembleDAgger uppvisar högre antal

avåkningar under testet. Även om samtliga algoritmer uppvisar stora variationer i resultat från iteration till iteration består skillnaden dem emellan.

Sammanfattningsvis kan man säga att RG-DAgger har en relativt bättre prestanda och vinner över de två andra DAgger-algoritmerna både i experttid och antal avkörningar, men vidare experiment behöver genomföras för att säkerställa resultaten.

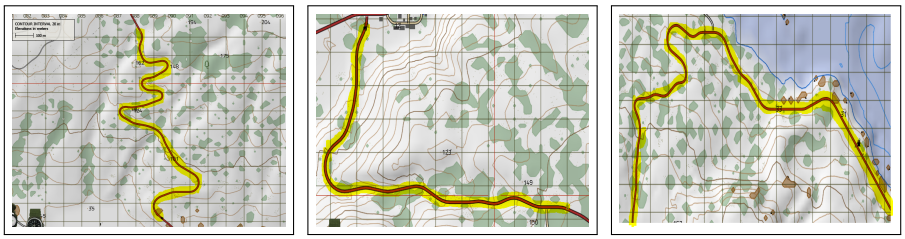
3.4 Experiment med stegvis imitationsinlärning

Vi föreslog en förbättring av DAgger-algoritmen som använder sig av EWC (elastisk viktkonsolidering) och gör aggregering av gamla data överflödig (se EWC-SD-algoritmen i avsnitt 2.4). I detta avsnitt redogör vi för de experiment som vi har utfört för att testa metoden.

EWC-SD-algoritmen utvärderades i kontexten självkörande fordon på tre banor i träningssimulatorn VBS3, dvs. två träningsbanor, 1 respektive 2, och en testbana (figur 3.4). Den initiala datauppsättningen som användes för att träna basmodellen samlades från träningsbana 1. All annan data samlades in från antingen träningsbana 1 eller 2. Testbanan användes endast för utvärdering av metoderna och data samlades aldrig in från den.

Tre experimentgenomföranden (körning 1, 2 och 3 i tabellerna) genomfördes för SafeDAgger och EWC-SD vardera i VBS3 för att utvärdera dess prestanda. Varje iteration av respektive genomförande innehöll körning på både träningsbana 1 och 2. Vidare utvärderades metoderna i varje iteration genom att mäta den avklarade sträckan i meter och om agenten lyckats slutföra banan. Varje beteendemodell fick köra tills den antingen nådde målet, körde av vägen eller körde in i motsatt körfält.

Experimenten visade att EWC-SD tränad enbart på nya data inte uppnår prestanda likvärdig SafeDAgger [18], men det räcker att en liten mängd träningsexempel (så kallad repeteringsbuffert) från gamla träningsdata läggs till för att EWC-SD skall överträffa SafeDAgger genom att uppnå likvärdig pre-



(a) Träningsbana 1 (b) Träningsbana 2 (c) Testbana

Figur 3.4 – Fågelperspektiv över träningsbanorna 1 (figur 3.4a), 2 (figur 3.4b) och testbanan (figur 3.4c). De röda kurvorna är vägar. Vägsträckor som har använts är markerat gult (bilden är hämtad från [18]).

standa i hälften så många iterationer. Samtliga tre körningar med SafeDAgger klarar testbanan efter att ha tränats i fyra iterationer, tabell 3.1, medan alla tre körningar med EWC-SD klarar testbanan efter endast två iterationer, tabell 3.2.

Slutsatsen är att EWC-SD-algoritmen med en repeteringsbuffert med endast få träningsexempel från gamla data löser problemen med ökande träningstider, ökande minneskrav samt kravet att alla tidigare data ständigt är tillgängliga som påtvingas av DAgger-algoritmen.

Tabell 3.1 – Resultat för SafeDAgger [18]

	Träningsbana 1		Träningsbana 2		Testbana	
	Sträcka (m)	Avklarad	Sträcka (m)	Avklarad	Sträcka (m)	Avklarad
Iteration 1						
Körning 1	1929.4	ja	1448.4	ja	502.7	nej
Körning 2	1929	ja	1448.1	ja	1888.1	nej
Körning 3	1930.2	ja	1448.1	ja	1472.9	nej
Iteration 2						
Körning 1	1927.1	ja	1449.6	ja	1552.6	nej
Körning 2	1926.6	ja	1447.1	ja	1941.9	ja
Körning 3	1927.8	ja	1449.6	ja	621.3	nej
Iteration 3						
Körning 1	1927.3	ja	1448.1	ja	204.4	nej
Körning 2	1929.8	ja	1449.6	ja	204.5	nej
Körning 3	1928.4	ja	1449.2	ja	204.4	nej
Iteration 4						
Körning 1	1928.4	ja	1446.5	ja	1941.6	ja
Körning 2	1927.1	ja	1447.5	ja	1940.9	ja
Körning 3	1926.8	ja	1448	ja	1939.9	ja

Tabell 3.2 – Resultat för EWC-SD med repeteringsbuffert [18]

	Träningsbana 1		Träningsbana 2		Testbana	
	Sträcka (m)	Avklarad	Sträcka (m)	Avklarad	Sträcka (m)	Avklarad
Iteration 1						
Körning 1	1929.2	ja	834.4	nej	780.9	nej
Körning 2	1929	ja	834.2	nej	1175.2	nej
Körning 3	1928.9	ja	832.9	nej	1176.9	nej
Iteration 2						
Körning 1	1928.7	ja	1445.6	ja	1939.9	ja
Körning 2	1926.3	ja	1445.5	ja	1939.1	ja
Körning 3	1927.2	ja	1445.5	ja	1938.1	ja

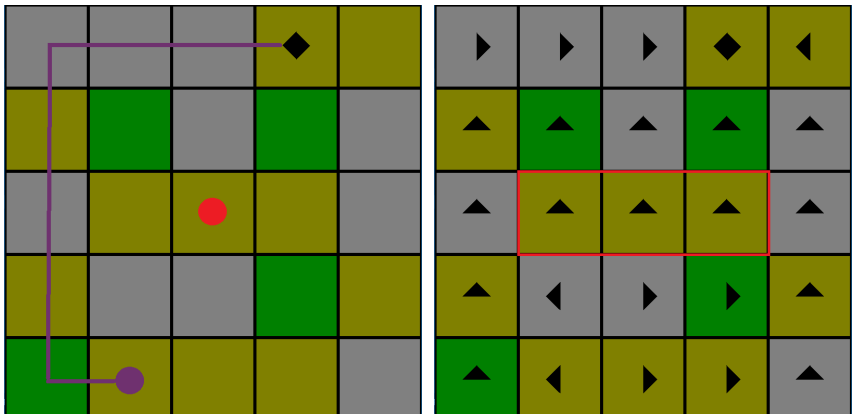
3.5 Experiment med inverterad förstärkt inlärning

I det här avsnittet redogörs för experiment med inverterad förstärkt inlärning (avsnitt 2.5), en familj av metoder för att imitera ett långsiktigt, målinriktat

beteende.

Experimenten undersökte möjligheten att använda inverterad förstärkt inlärning till att lära en agent taktisk ruttplanering utifrån exempel på taktiskt planerade rutter; experimenten antar en expert som kan demonstrera taktiskt lämpliga rutter men som inte kan formalisera de underliggande principerna (som kommer till utryck i expertens val av rutter). Expertens ruttplanering är taktisk i det att experten värderar en position på kartan utifrån dess relation till andra positioner och terrängen och objekten där. Agenten kan exempelvis värdera risken för upptäckt i en kartposition utifrån terrängen längs med siktlinjer till fiender.

Det första experimentet genomfördes i en abstrakt terräng (figur 3.5) som är en modifiering av *object world* som används flitigt som benchmark i litteraturen [28]. Kartan har modellerats som ett rutnät av 10 x 10 rutor som slumpmässigt har tagits fram (figur 3.5a visar ett mindre exempel på en 5 x 5 karta). Varje ruta tillhör en av tre olika terrängtyper, öppet fält (ljusgrön), skog (grön) och berg (grå) och den enda egenskapen som skiljer mellan dessa rutor är att de påverkar siktförhållandet. En agent (den lila pucken) och en



(a) Träningsdata (karta och rutt) (b) Inlärd policy

Figur 3.5 – Exempel på terräng som användes i det första experimentet med inverterad förstärkt inlärning [27]. Bilden till vänster (a) visar agenten (den lila pucken), destinationen (den svarta fyrkanten), den fientliga observatören (den röda pucken) och en rutt vald av en expert (den lila rutten). Bilden till höger (b) visar den policy som har lärts in. Riktningen på den svarta pilen i varje ruta bestämmer den riktning agenten ska röra sig i från respektive ruta. Det finns inga pilar som pekar mot den öppna terrängen i mitten av kartan som har markerats med röda linjer, vilket innebär att agenten undviker detta område och håller sig till den mer täckande terrängen på kanterna som ger mer skydd.

destination (den svarta fyrkanten) placeras slumpmässigt på två rutor i terrängen och agenten får uppgiften att flytta sig till destinationen samtidigt som den ska minimera risken för att visuellt bli upptäckt av en fientligt observatör som också har placerats slumpmässigt på kartan (den röda pucken). Flyttning är tillåten endast till rutor som har en angränsande sida. Vid start har agenten tillgång till själva kartan, dvs. terrängtypen för respektive ruta, men har ingen kännedom om hur dessa ska tolkas och hur de påverkar siktförhållandet och risken att bli upptäckt. Däremot får den exempeldata i form av en rutt som en expert har valt för att flytta (den lila rutten). Inlärningsalgoritmen ska estimera den okända, icke-linjära kostnadsfunktionen utifrån en uppsättning exempeldata dvs. kartor och respektive rutter som minimerar den okända kostnadsfunktionen (upptäcktsrisken). Kostnadsfunktionen används för att ta fram policyn, alltså beskrivningen av agentens beteende (figur 3.5b).

I detta experiment prövades Maximum Entropy Deep Inverse Reinforcement Learning [28], en nyare variant av Maximum Entropy Inverse Reinforcement Learning [23] vilken kan sägas utgöra state-of-the-art inom området. Med dessa metoder estimeras den okända kostnadsfunktionen genom en serie gissningar som successivt konvergerar till en lösning; för att generera nästa gissning i serien görs en kostsam optimering.

Våra experiment visade att inverterad förstärkt inlärning har potentialen att lösa problemet med taktiskt ruttplanering men algoritmen kräver stor beräkningskraft. Med en standarddator skalar inte metoden till kartor mycket större än 10 x 10 rutor på grund av just de kostsamma optimeringarna [27].

I ett andra experiment, som genomförts i ett exjobb [29] inom ramen för detta arbete, vilket utfördes på object world (utan någon modifiering) prövades metoder att snabba upp optimeringsstegen genom att överföra information från det föregående optimeringssteget; varje optimeringsproblem i serien är väldigt likt det föregående optimeringsproblem, men denna information tas inte alls till vara utan varje optimering i serien sker tabula rasa. Experimenten bekräftade att den inverterade förstärkta inlärningen skalade bättre med informationsöverföring mellan optimeringsstegen, men att effekten varierade mycket. Ibland gick inlärningen många storleksordningar snabbare, andra gånger endast försumbart snabbare. För mer information om object world och vår metod se [28, 29].

4 Diskussion

I denna rapport studerade vi hur imitationsinlärning kan användas för beteendemodellering i virtuella miljöer och simulatorer. Rapporten beskriver olika sätt för en agent att lära sig att utföra en uppgift genom att träna på exempeldata som produceras när en (mänsklig) expert utför uppgiften.

Vi började med beteendekloning, en form av övervakad inlärning som är förhållandevis enkel att tillämpa. Resultaten visade att beteendekloning tenderar att ge ett instabilt beteende, men kanske ändå tillräckligt stabilt för vissa tillämpningar inom simuleringsbaserad övning och träning där spelledningen lätt kan gå in och korrigera ett misstag (så som att flytta ett självkörande fordon som kört fel).

Det instabila beteendet beror på att övervakad inlärning, för att fungera väl, behöver tränings- och testdata som är oberoende och likafördelade, vilket i allmänhet inte blir fallet när en agent (beteendemodell) interagerar med en omgivning (simulator) i en sluten slinga.

Interaktiv imitationsinlärning (dvs. olika varianter av DAgger-algoritmen) försöker åstadkomma mer likafördelad tränings- och testdata genom att låta agenten (beteendemodellen) interagera med omgivningen en stund varpå kontrollen lämnas över till den mänskliga experten och ny data samlas in och aggregeras med den tidigare, varefter beteendemodellen tränas om på den aggregerade datamängden; den nya beteendemodellen får sedan interagera med omgivningen en stund varpå återigen kontrollen lämnas över till experten, osv. i ett antal iterationer. De olika varianterna av DAgger skiljer sig framförallt i när och hur kontrollen växlar mellan beteendemodell och expert.

DAgger är mer robust och ger möjligheten att förfina modellen successivt, men ställer större krav på datainsamlingen – den mänskliga experten som ska efterliknas måste vara tillgänglig och interagera med systemet vid felstyrning och producera mer träningsdata. Vilken variant av DAgger som är mest lämplig för en viss simulator beror på olika faktorer, men med stor sannolikhet kommer DAgger att vara en delkomponent av varje datadriven beteendemodellering för agenter i simulatorer.

Vi introducerade även två utökningar av DAgger: dels belöningsorienterad imitationsinlärning (RG-DAgger) och dels stegvis imitationsinlärning (EWC-SD), en utökning av SafeDAgger med elastisk viktkonsolidering. I RG-DAgger används en tillämpningsspecifik heuristik för att besluta om när under träningen beteendemodellen ska lämna över kontrollen till den mänskliga experten. Trots att RG-DAgger tillhandahåller relativt bättre resultat jämfört med de etablerade varianterna av DAgger, är den i dess nuvarande implementation beroende av domänspecifik information som explicit måste extraheras från simulatorn.

Med EWC-SD tränas modellen inte på hela den aggregerade datamängden

utan enbart på ny expertdata i varje iteration, varvid elastisk viktkonsolidering används för att skydda modellens tidigare inlärda kunskaper mot katastrofal glömska. EWC-SD är kanske det mest intressanta och lovande experimentet, eftersom den även gör det möjligt att träna en modell för nya uppgifter.

De olika metoderna för imitationsinlärning som diskuteras i rapporten är alla end-to-end: Beteendemodellen mappar indata (tillstånd) till utdata (handlingar) och kringgår de mellanliggande stegen som vanligtvis finns i en traditionell agentarkitektur. Rapportens beteendemodeller för bilförare bestämmer exempelvis styrsignal direkt utifrån förarvyn. Det behövs därmed ingen kännedom om hur simulatorn fungerar internt. Det är exempelvis inte nödvändigt att förstå hur ett fordon behöver svänga ett visst antal grader åt höger givet ett visst tillstånd för att därifrån beräkna den nödvändiga handlingen som krävs för att åstadkomma denna effekt, utan modellen ger direkt utslaget på styrspaken. Undantaget är träningsfasen för RG-DAgger som bygger på att man inspekterar simulatorns inre tillstånd i beslutsregeln. Även om detta är en domän- och situationsspecifik anpassning är det endast ett hjälpmedel som underlättar inlärningen, den färdigtränade agenten handlar utan hjälp och endast utifrån bilder ur förarvyn.

Imitationsinlärningsmetoder som är end-to-end är i allmänhet inte bundna till en viss simulator. Om en metod fungerar bra i en simulator kan den mer eller mindre direkt överföras och användas i en annan simulator. Till exempel används beteendekloning och DAgger-algoritmen – som här har använts för styrning av fordon i VBS3 – på försök i ett annat projekt för styrning av stridsflygplan hos FLSC [7].

I denna rapport utgörs expertagenten i DAgger av en mänsklig operatör som novisagenten försöker att efterlikna. Men DAgger används även med automatiska expertagenter, bl.a. i den banbrytande algoritmen AlphaZero [3, 5] i vilken expertagenten utgörs av en trädsökningsalgoritm.

Vi har studerat ett brett spektrum av imitationsinlärningsmetoder, från beteendekloning till olika DAgger-algoritmer, en kombination av stegvisinlärning med DAgger, och inverterad förstärkt inlärning, samtliga med fokus på modellering av datorgenererade styrkor. Arbetet har bidragit med studier, implementering, experiment och utvärdering av ledande imitationsinlärningsmetoder. Även ett antal vidareutvecklingar utforskades, däribland följande två.

- En hybrid metod där imitationsinlärning kombineras med en belöningssignal i syfte att effektivisera träningen (RG-DAgger). I en icke uttömmande utvärdering visade sig metoden ha bättre prestanda jämfört med etablerade DAgger-algoritmer.
- En metod som kombinerar stegvis inlärning med DAgger så att det går att anpassa (träna om) modellen till nya omgivningar oberoende av äldre data (EWC-SD). Metoden har även potentialen att användas för att träna agenter på nya uppgifter, detta har dock inte studerats. Anpassning till nya omgivningar och träning på nya uppgifter är två centrala aspekter för beteendemodellering av agenter som EWC-SD adresserar.

5 Slutsatser

Under de senaste fem åren har beteendemodellering med maskininlärning gått från att bara kunna hantera leksaksexempel till att bemästra väldigt komplicerade spel så som StarCraft II, där många olika agenters beteende styrs samtidigt i realtid i en mycket komplex omgivning där beslutsfattaren har begränsad information om tillståndet.

AlphaStar, algoritmen som användes i fallet med StarCraft II, bygger på att agenterna initialt tränas med imitationsinlärning för att sedan tränas vidare med de andra AI-metoderna förstärkt inlärning och multi-agent inlärning [4]. Även om AlphaStar, i likhet med många andra framgångssagor de senaste åren, bygger på en sinnrik sammankoppling av olika maskininlärningsalgoritmer, är det värt att observera att trenden går mot allt mer generella metoder och dellösningar där allt mindre domänkunskap byggs in i algoritmerna.

Imitationsinlärning är ett ungt och spretigt forskningsområde, där mycket ännu är outrett. Framgångar på området kommer att få stor betydelse när man kan överföra förmåga från människa till maskin i större omfattning.

På FOI jobbar vi med kunskapsuppbyggande inom imitations- och förstärkt inlärning, i ett kort och ett långt perspektiv. I det korta perspektivet handlar det om att utforska om och hur det går att ta fram agenter för olika simuleringsverktyg, vilket medför större potential att genomföra och utvärdera stora simuleringar av förlopp med hög realism. I det långa perspektivet handlar det om att stå redo med kompetensen, ifall förstärkt inlärning och imitationsinlärning blir avgörande för en potent förmåga i moderna försvars- och säkerhetssystem.

Chefen för Joint Artificial Intelligence Center under Pentagon, Jack Shanahan, uttryckte vikten av att hänga med i AI-forskningen: “vad jag inte vill se är en framtid där våra motståndare har en fullt AI-kapabel styrka, som vi själva saknar”, i en intervju till the Economist [30].

Litteraturförteckning

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[2] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[3] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550:354–359, 2017.

[4] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575:350–354, 2019.

[5] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[6] Erik Båvenstrand and Jakob Berggren. Performance evaluation of imitation learning algorithms with human experts, B.S. Thesis, KTH Royal Institute of Technology, 2019.

[7] Linus J. Luotsinen, Andreas Horndahl, and Emil Larsson. *Statusrapport – Maskininlärningsförmåga i FLSC.* FOI Memo 6634, Totalförsvarets forskningsinstitut, 2018.

[8] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 7(1-2):1–179, Mar 2018.

[9] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[10] Stephane Ross and Drew Bagnell. Efficient reductions for imitation learning. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Machine Learning Research*, pages 661–668, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[11] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[12] Kunal Menda, Katherine R. Driggs-Campbell, and Mykel J. Kochenderfer. EnsembleDAgger: a Bayesian approach to safe imitation learning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[13] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end simulated driving. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4–9, 2017, San Francisco, California, USA.*, pages 2891–2897. AAAI Press, 2017.

[14] Nora Al-Naami. Imitation learning using reward-guided dataset aggregation (RG-DAgger). Master's thesis, KTH Royal Institute of Technology, 2019.

[15] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.

[16] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3:128–135, 1999.

[17] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[18] Andreas Elers. Continual imitation learning: Enhancing safe data set aggregation with elastic weight consolidation. Master's thesis, KTH Royal Institute of Technology, 2019.

[19] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[20] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 2. MIT press Cambridge, 1998.

[21] Bilal Piot, Matthieu Geist, and Olivier Pietquin. Bridging the gap between imitation learning and inverse reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 28(8):1814–1826, 2017.

[22] Per-Idar Evensen and Dan Helge Bentse. *Simulation of land force operations – a survey of methods and tools.* FFI-rapport 2015/01579, Forsvarets forskningsinstitutt, 2016.

[23] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pages 1433–1438. AAAI Press, 2008.

[24] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620–630, 1957.

[25] Linus J. Luotsinen and Farzad Kamrani. *Slutrapport, syntetiska aktörer: datadriven beteendemodellering.* FOI-R--4513--SE, Totalförsvarets forskningsinstitut, 2017.

[26] Farzad Kamrani, Mika Cohen, and Emil Larsson. *Lagom intelligenta datorgenererade styrkor.* FOI Memo 6587, Totalförsvarets forskningsinstitut, 2018.

[27] Emil Widham. Tactical route planning in battlefield simulations with inverse reinforcement learning, B.S. Thesis, KTH Royal Institute of Technology, 2019.

[28] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning, 2015. arXiv:1507.04888 [cs.LG].

[29] Emil Widham. Scaling up maximum entropy deep inverse reinforcement learning with transfer learning. Master's thesis, KTH Royal Institute of Technology, 2019.

[30] Battle algorithm. *The Economist*, page 67–69, Sep 2019.

FOI är en huvudsakligen uppdragsfinansierad myndighet under Försvarsdepartementet. Kärnverksamheten är forskning, metod- och teknikutveckling till nytta för försvar och säkerhet. Organisationen har cirka 1000 anställda varav ungefär 800 är forskare. Detta gör organisationen till Sveriges största forskningsinstitut. FOI ger kunderna tillgång till ledande expertis inom ett stort antal tillämpningsområden såsom säkerhetspolitiska studier och analyser inom försvar och säkerhet, bedömning av olika typer av hot, system för ledning och hantering av kriser, skydd mot och hantering av farliga ämnen, IT-säkerhet och nya sensorers möjligheter.

FOI
Totalförsvarets forskningsinstitut
164 90 Stockholm

Tel: 08-55 50 30 00
Fax: 08-55 50 31 00

www.foi.se