Självlärande Hare and Hounds-spelare med Q-learning

HARALD HARTWIG
och MAX WESTERMARK

KTH Datavetenskap
och kommunikation

Examensarbete
Stockholm, Sverige 2011

Självlärande Hare and Hounds-spelare med Q-learning

H A R A L D H A R T W I G
o c h M A X W E S T E R M A R K

Examensarbete i datalogi om 15 högskolepoäng
vid Programmet för datateknik
Kungliga Tekniska Högskolan år 2011
Handledare på CSC var Lars Kjelldahl
Examinator var Mads Dam

URL: www.csc.kth.se/utbildning/kandidatexjobb/datateknik/2011/hartwig_harald_OCH_westermark_max_K11048.pdf

Kungliga tekniska högskolan
Skolan för datavetenskap och kommunikation

KTH CSC
100 44 Stockholm

URL: www.kth.se/csc

Referat

Denna rapport behandlar belöningsbaserad inlärning, en gren inom maskininlärning. I rapporten undersöks huruvida en Q-learning agent kan lära sig att spela brädspelet Hare and Hounds. Genom att implementera Q-learning har analyser kunnat genomföras på olika aspekter av inlärningsprocessen. De experiment som genomförts för att undersöka Q-learning är ett test på inlärningsparametrarna, ett test mot en simpel spelstrategi samt ett test som visar Q-learnings konvergens. Undersökningarna visar att Q-learning lämpar sig väl för att lära sig spela Hare and Hounds.

Abstract

Self Learning Hare and Hounds player with Q-learning

This report deals with reinforcement learning, a branch of machine learning. The report examines whether a Q-learning agent can learn to play the board game Hare and Hounds. By implementing the Q-learning algorithm, analysis has been completed on different aspects of the learning process. The experiments performed to examine the Q-learning is one test of the learning parameters, one test against a simple strategy and one test that shows the Q-learning convergence. The investigations show that Q-learning is well suited for learning to play the board game Hare and Hounds.

Förord

Detta är en kandidatexamensrapport för kursen DD143X genomförd vårterminen 2011 på KTH, Stockholm. Rapporten och det bakomliggande arbetet genomfördes i samarbete mellan båda författarna som bidrog med lika stora delar.

Tack till vår handledare Lars Kjelldahl för rådgivning under projektets gång. Ett tack ges även till Johan Boye för vägledning som förde projektet framåt.

Innehåll

Kapitel 1

Inledning

Artificiell Intelligens är ett stort forskningsområde inom datalogi som ständigt växer. Det finns idag enorma mängder program som kan integrera med sin omgivning på bästa sätt genom att ha lärt sig vad som är rätt. Enkla brädspel är ett utmärkt sätt för en agent att lära sig, eftersom spelen har väldefinierade regler, bestämda mål och inte är för komplexa. Vår undersökning inriktar sig på att implementera en självlärande agent och studera dess lärandeprocess.

1.1 Problemformulering

Kan en agent som använder sig utav Q-learning lära sig att spela strategibrädspelet Hare and Hounds? Vilka värden på parametrarna epsilon, alpha och gamma är de optimala vid inlärning? Kommer agenten, efter inlärning, att klara sig mot en mänsklig motståndare?

Kapitel 2

Teori och Bakgrund

2.1 Maskininlärning

Maskininlärning är en gren inom artificiell intelligens som handlar om design och utveckling av algoritmer som tillåter datorer att lära sig utifrån erfarenhetsmässig data. Inom maskininlärning finns det flera olika algoritmer för att implementera en inlärningsprocess. En av dessa är belöningsbaserad inlärning, *reinforcement learning*.[1]

2.1.1 Belöningsbaserad inlärning

Belöningsbaserad inlärning bygger på att programmet själv lär sig hur det ska agera i olika situationer genom att testa möjliga handlingar och observera resultatet. Det långsiktiga målet med belöningsbaserad inlärning är, precis som namnet insinuerar, att programmet ska få strategiska kunskaper inom dess handlingsmiljö genom att maximera en belöningsfunktion på lång sikt. Q-learning är en metod inom belöningsbaserad inlärning.[5]

2.1.2 Optimal Control och Temporal Difference

Optimal Control och *Temporal Difference learning*, TD-learning, är två metoder inom belöningsbaserad inlärning. Optimal Control handlar om att maximera eller minimera en given kostnadsfunktion. Genom att maximera, eller minimera, ett problem som kan beskrivas i en kostnadsfunktionen, kan man hitta ett optimalt tillvägagångssätt på det givna problemet. Ett enkelt exempel är hur man ska ta sig till en given plats på kortast tid, där kostnadsfunktionen utgörs av variabler som hastighet, hinder på vägen och avstånd.[5]

Temporal Difference learning, eller TD-learning, bygger på att man förutser framtida händelser beroende på tidigare händelser i en given modell. Beroende på vad man har upplevt, och därmed lärt sig, kan TD-learning göra kvalificerade gissningar på hur framtida händelser bör se ut.[5]

Optimal Control och Temporal Difference utgör grunden för Q-learning. 1989 förenade Chris Watkins dessa två metoder till vad som skulle komma att bli Q-learning.[6]

2.1.3 Q-learning

Q-learning är en inlärningsalgoritm som delar upp problemet i tillstånd, där varje tillstånd har givna handlingar. Handlingarna transporterar agenten mellan tillstånden. En betydande styrka med Q-learning är att man endast behöver definiera belöningen för sluttillstånden. Bästa vägen dit beräknas av algoritmen.

Q-learning fungerar endast om uppgiften som ska lösas kan beskrivas som en *Markoviansk beslutsprocess*. Det innebär att uppgiften utgörs av en sluten tillståndsrymd, det vill säga om antalet tillstånd är ändligt. Eftersom varje handling leder till ett nytt tillstånd och tillståndsrymden är ändlig, innebär det även att handlingsrymden är ändlig. Varje handling mellan två tillstånd har alltid en belöning, som avgör vilken handling som är bäst i ett givet tillstånd.[6]

För att hitta den optimala handlingen i ett givet tillstånd utnyttjar Q-learning en värdefunktion. Värdefunktionen ger ett värde för alla handlingar i alla tillstånd, beroende på den förmodade framtida belöningen. Q-learnings syfte är att lära sig denna värdefunktion. Den optimala handlingen fås när värdefunktionen är som störst. Den optimala handlingen kallas för *optimal policy*.

Då värdefunktionen är okänd approximerar Q-learning värdefunktionen med vad som kallas Q-funktionen. Q-funktionen, även kallad Q-värde, uppdateras då agenten har utfört en handling. Värdet för varje handling i ett givet tillstånd lagras som Q(s,a), där s är tillstånd(eng. state) och a är handling(eng. action). Q-funktionen uppdateras enligt:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left(R(s_{t+1}) + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \tag{2.1}$$

I formeln är α inlärningshastighet(eng. learning rate). Funktionen R(s_{t+1}) är belöningen för att komma till tillstånd s_{t+1}, där s_{t+1} är tillståndet man når genom att utföra handling a_t i tillstånd s_t. Parametern γ är en faktor som viktar framtida belöningar. Funktionen $maxQ(s_{t+1}$,a) är det största Q-värdet för alla handlingar i nästa tillstånd. Förutom α och γ används även en tredje parameter, ϵ.[4]

Epsilon, ϵ

Parametern ϵ är den parameter som bestämmer om agenten ska använda den bästa inlärda handlingen vid ett visst tillstånd, eller om agenten ska välja en handling slumpmässigt. Epsilon kan anta värdena:

$$0 \leq \epsilon \leq 1 \tag{2.2}$$

Då ϵ är 1 väljs alla handlingar slumpmässigt och om ϵ är 0 väljer den enbart inlärda handlingar.

Alpha, α

Parametern α bestämmer inlärningshastigheten hos agenten. För värden närmare 0 kommer ny information att få mindre vikt och för värden närmare 1 kommer ny information skriva över gammal information. Alpha kan anta värdena:

$$0 \leq \alpha \leq 1 \tag{2.3}$$

Gamma, γ

Parametern γ bestämmer hur agenten ska värdera framtida belöningar. För värden närmare 0 kommer den bara söka kortsiktig belöning. Vid värden närmare 1 kommer agenten stäva efter ett så högt framtida mål som möjligt. Gamma kan anta värdena:

$$0 \leq \gamma \leq 1 \tag{2.4}$$

Då samtliga parametrar är 0 lär sig agenten ingenting och använder sig endast det den har lärt sig tidigare.

Q-learnings konvergens

Styrkan med Q-learning är att Q-funktionen konvergerar, vilket bevisades 1992 av C. Watkins och P. Dayan.[6] Att den konvergerar innebär att Q-funktionen inte längre uppdateras, och därmed är en god approximation till värdefunktionen. Då Q-funktionen har konvergerat har en optimal policy hittats.

2.1.4 Tidigare forskning

Många brädspel, även om de anses som barnspel, är ett sätt för forskare att testa nya metoder inom maskininlärning. Eftersom spelreglerna oftast är enkla är de lätta att implementera men spelen kan oftast innehålla komplexa strategier. Brädspelen kan också vara förenklingar av problem som inträffar i verkliga livet.[3]

Det finns även ett flertal kandidatexamensrapporter från KTH som handlar om Q-learning applicerat på brädspel. Dessa rapporter är från 2010, alltså förra årets examensarbete inom datalogi på grundnivå.

2.2 Hare and Hounds

Hare and Hounds är ett strategiskt brädspel för två personer. Det finns flera olika varianter av spelet och dess samlingsnamn är Harspel. Dessa spel var populära under medeltiden i Europa fram till 1900-talet. Även om spelbrädet och startplacering kan variera mellan de olika varianterna är grundreglerna desamma. Spelaren med 3 pjäser, oftast gestaltade av hundar, ska fånga spelaren med 1 pjäs, oftast gestaltad av en hare, samtidigt som harspelaren ska rymma, vilket han gör då han passerat samtliga hundar sett från vilken sida av spelplanen haren startat på.[8]

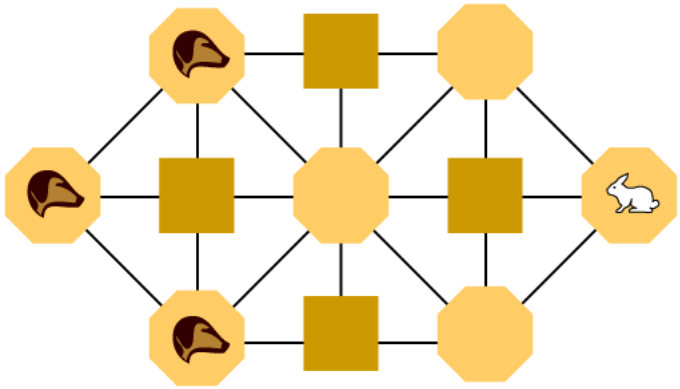
Just Hare and Hounds, även känt som *The French Military Game*, var populärt bland officerarna i den franska armén under fransk-tyska(fransk-preussiska) kriget som utkämpades 1870-1871. Spelet är ett typexempel på spel som studeras i kombinatorisk spelteori.[7] Matematikern Martin Gardner sa:

"Hare and Hounds combines extreme simplicity with extraordinary strategic subtlety"[2]

Fritt översatt:

"Hare and Hounds kombinerar extrem enkelhet med extraordinär strategisk briljans”

i sin kolumn, Matematiska Spel, i tidningen Scientific American.[2] Spelet bör därför lämpa sig väl för att undersöka huruvida en Q-learning agent kan lära sig att spela det.



Figur 2.1. Hare and Hounds spelplan. Pjäserna står i startposition.

2.2.1 Analyserande data

Hare and Hounds är ett väldigt litet spel, trots det faktum att det finns flera varianter på det. Mycket tid spenderades åt sökande av analyserande data, till exempel algoritmer av vinnande strategi eller vilken spelare som har störst chans till vinst då båda spelarna spelas på bästa sätt. Dessvärre fanns ingen sådan data tillgänglig.

Arbetet kring implementationen skulle gått betydligt enklare om sådan data hade funnits tillhanda då bättre träningsspelare hade kunnat implementerats på ett enklare sätt. Även de analytiska slutsatserna av Q-learnings framgång på Hare and Hounds hade kunnat diskuterats på ett bättre och mer korrekt vis.

2.2.2 Spelregler

- En spelare representerar de tre hundarna som ska försöka stänga in den andra spelarens hare samtidigt som denne försöker undkomma.
- Varje spelare kan gå ett steg på sin tur. Hundarna kan bara gå åt höger, diagonalt åt höger eller vertikalt, men aldrig åt vänster. Haren kan gå i alla riktningar.
- Hundarna vinner om de ”stänger in” haren så att denne ej längre kan gå någonstans.
- Haren vinner om den kan fly från hundarna. Att fly innebär att haren är till vänster om alla hundarna.
- Om hundarna rör sig vertikalt tio steg i rad anses det som fördröjning och haren vinner.

Källa: [7]

Kapitel 3

Implementation

För att kunna besvara frågeställningen har en implementation av Hare and Hounds programmerats i java. Vi har även byggt en hundspelare som använder Q-learning som studeras i våra analyser. För att hundspelaren ska kunna lära sig har två harspelare även implementerats. En helt slumpmässig och en med en simpel strategi.

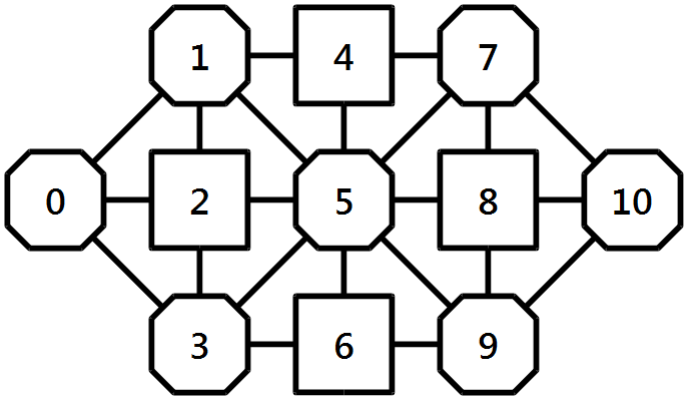
Motivering av implementation

Anledningen till att implementationen skrevs i Java var av skälet att vi, författarna, kunde Java bättre än andra programmeringsspråk som skulle kunna ha implementerats på ett liknande sätt. De tilltänkta programmeringsspråk som valdes bort på grund av svagare kunskaper var C++ och Python.

Valet att låta hundspelaren lära sig spelet Hare and Hounds genom Q-learning istället för harspelaren bygger på det sätt vi valt att träna upp vår agent. Eftersom det inte finns någon tillgänglig algoritm hur spelarna ska spela för att vinna var vi tvungna att själva programmera våra träningsspelare, och valet föll då på att harspelaren skulle agera träningsspelare. Detta val byggdes främst på det faktum att harspelaren endast styr en pjäs, medan hundspelaren styr tre. Det skulle bli mer komplicerat att göra en träningsspelare med hundspelaren, men inte avsevärt mer komplicerat att implementera Q-learning på harspelaren.

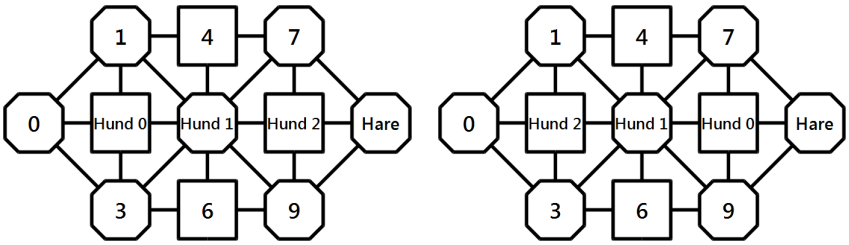
3.1 Spelplan och spellogik

Spelplanen är implementerad som en graf där varje spelruta är en nod i grafen. Kanter representeras av vägarna mellan spelrutorna. För att på ett effektivt sätt kontrollera vilken position hundarna och haren befinner sig på, är spelplanen indexerad, se figur 3.1.



Figur 3.1. Indexerad spelplan

Ett tillstånd i spelet ges av var på spelplanen de tre hundarna står, haren står och vad hundarnas fördröjningsvärde är. Hundarna är numrerade från 0 till 2. Detta leder till att kommande två tillstånd är olika när dom bör vara samma. På grund av detta sorterar vi hundarnas positioner efter varje drag. Sorteringen går till på så vis att den hund som står på rutan med högst index blir hund nr 2, den hund som står på lägst index blir hund nr 1.



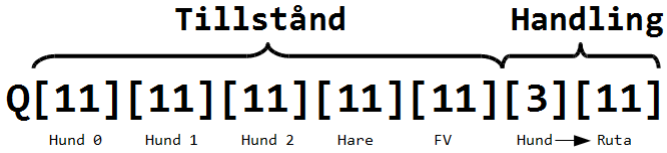
Figur 3.2. Med sortering blir dessa två tillstånd samma.

3.2 Q-Learning Hundspelare

Q-learning hundspelaren är implementerad på så vis att man i början av varje spelomgång får välja vilka värden parametrarna α, γ och ϵ ska anta. Väljer man värdet 0 på ϵ när agenten är olärd, dvs alla Q-värden är 0, då slumpas en handling fram utifrån de handlingar som är möjliga för hundarna i det tillståndet. Detsamma gäller då agenten har blivit upplärd och två eller fler Q-värden har antagit samma högsta värde. Det vill säga att två eller fler handlingar är exakt lika bra i det tillståndet.

3.2.1 Q-funktion

Q-funktionen är representerad av en vektor med följande dimensioner:



Figur 3.3. Visar vad som är vad i indexeringen i Q-vektorn. FV står för fördröjningsvärde.

De fyra första och det sista indexet, Q[11][11][11][11][][][11], är av storlek 11 eftersom spelplanen har 11 spelrutor, och varje hund och hare kan vara placerad på samtliga spelrutor i något givet tillstånd. De fem första indexen motsvarar tillståndet, s i Q(s,a), varav de 3 första betecknar hundarnas position på spelplanen. Den fjärde är harens position och den femte är hundarnas nuvarande fördröjningsvärde. De två sista indexen motsvarar en handling för det givna tillståndet där det första indexet av de två är vilken hund som ska gå. Det andra pekar ut till vilken ruta den valda hunden ska till. Q-vektorn har plats för Q-värden för tillstånd och handlingar som inte existerar, t.ex. att alla hundar står på samma ruta. Alla dessa tillstånd kommer aldrig besökas vilket medför att den verkliga tillståndsrymden är mindre än Q-vektorns kapacitet.

Exempel: Q[2][5][8][10][0][2][7] motsvarar Q-värdet för handlingen att hund 2 ska gå till spelruta 7, i den vänstra spelplanen i figur 3.2

3.2.2 Uppdatering av Q-värden

Efter varje utfört drag ska Q-värdet för handlingen som precis utfördes uppdateras med ett nytt värde enligt 2.1. Formeln fungerar dock bara när man vet till vilket tillstånd en utförd handling leder. I ett spel med två spelare som spelar vartannat drag vet man inte i vilket tillstånd man hamnar direkt efter sitt drag. Vi definierar därför ett drag som att både hundarna och haren har gjort varsitt drag.

Därefter uppdateras Q-värdet för handlingen i det förra draget. Uppdateringsfunktionen som används blir:

$$Q(s_{t-1}, a_{t-1}) = Q(s_{t-1}, a_{t-1}) + \alpha \left(R(s_t) + \gamma \cdot \max_a Q(s_t, a) - Q(s_{t-1}, a_{t-1}) \right) \quad (3.1)$$

Belöning

Det enda en olärd agent vet om Hare and Hounds är att den kan gå, och när den har vunnit eller förlorat. I de första spelomgångarna vet den inte ens huruvida nästa drag kommer att vara ett vinnande eller förlorande. Vid vinst får agenten 1 i belöning och vid bägge förlustsituationerna, att haren rymmer eller att hundarna fördröjer spelet, får agenten -1 i bestraffning.

3.2.3 Träningsspelare

Slumpmässig Harspelare

Denna harspelare väljer ett av sina möjliga drag helt slumpmässigt. Spelaren används som träningsmotståndare till Q-Learning hundspelaren. Eftersom den spelar helt slumpmässigt kommer man, om man spelar tillräckligt många spel, ha genomsökt hela tillståndsrymden. Därigenom kan vi genom träning mot denne få Q-learning algoritmen att konvergera.

Simpel Harspelare

En harspelare som alltid spelar med samma, mycket simpla, strategi. Den försöker alltid utföra det drag som är mest åt höger, sett från harens synvinkel. Med indexeringen av spelplanen innebär det att den alltid försöker gå till den ruta med lägst index, se figur 3.1. Används för att testa hur effektivt Q-learning hundspelaren kan lära sig att vinna mot en specifik strategi.

Motivering av träningsspelare

De två valda träningsspelarna fyller varsin funktion i undersökandet om Q-learning algoritmen kan implementeras på ett effektivt sätt på spelet Hare and Hounds. En ytterligare träningsspelare vid namn *Avancerad Harspelare* var från början inkluderad i planeringen av implementationen, men i brist på en effektiv spelalgoritm var detta inte genomförbart.

Ett intressant experiment skulle varit att implementera en Q-learning hundspelare som spelar mot en Q-learning harspelare, där de tränar mot varandra. På så vis, i teorin, skulle de lära sig att motverka varandras vinnande strategier. Detta experiment uteblev tyvärr i brist på tid då den övriga implementationen drog ut på tiden.

Kapitel 4

Analys och Resultat

Detta kapitel behandlar undersökningen av vilka parametrar som är de bästa vid inlärning, påvisar inlärning mot en simpel strategi och visar på att Q-tabellen konvergerar.

4.1 Optimering av inlärningsparametrar

Med att optimera inlärningen menar vi att hitta de värden på parametrarna α, γ, och ϵ då agenten både lär sig snabbt, mycket och stabilt.

Vid optimeringen av parametrarna har vi valt att träna agenten mot den slumpmässiga harspelaren. Detta eftersom agenten ska få besöka så många tillstånd som möjligt i tillståndsrymden. Slumpfaktorn leder till viss osäkerhet, men efter tillräckligt många spelomgångar kan den försummas. Agenten konvergerar ändå mot ett värde till slut. Dessutom är syftet inte exakt noggrannhet, utan syftet är att kunna urskilja ett samband i kombinationen mellan α och γ.

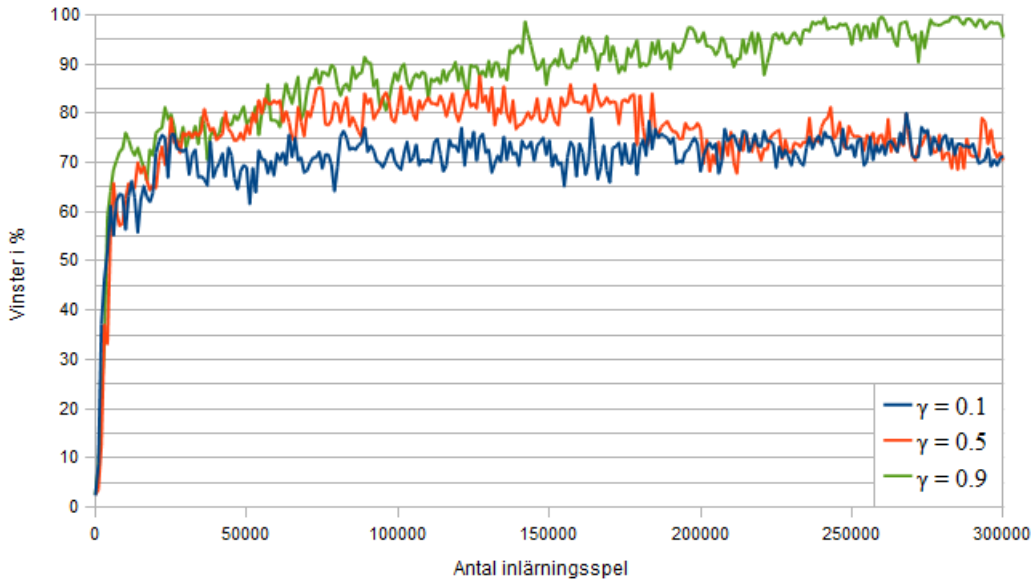
Samtliga grafer nedan, som analyserar värdena på α och γ, körs på samma sätt:

1. Först körs 300 inlärningsspel med angivna värden på α, γ och $\epsilon = 1.0$.

2. Sedan pausas inlärningen och 500 spelomgångar med samtliga inlärningsparametrar satta till 0, dvs agenten lär sig inget nytt och använder sig endast av tidigare inlärda kunskaper, körs. En vinstprocent av de 500 spelomgångarna räknas ut och visas i grafen.

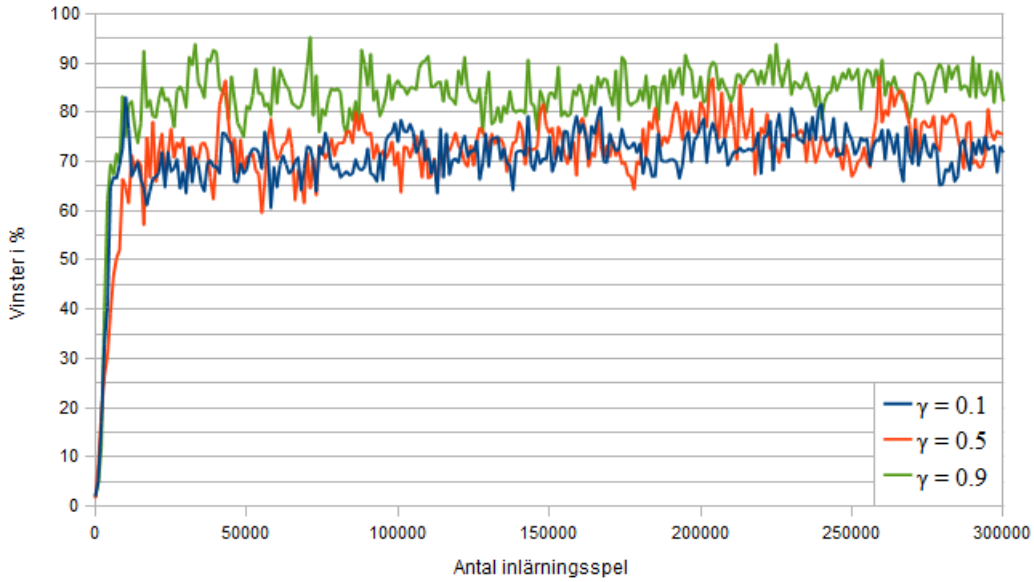
3. Efter 500 spelomgångar med ovan nämnda inställningar återupptas agentens inlärning. Värdena som visas i grafen är vinstprocenten för vart 300:e inlärningsspel av sammanlagt 300 000 spelomgångar.

4.1.1 Gamma och alpha



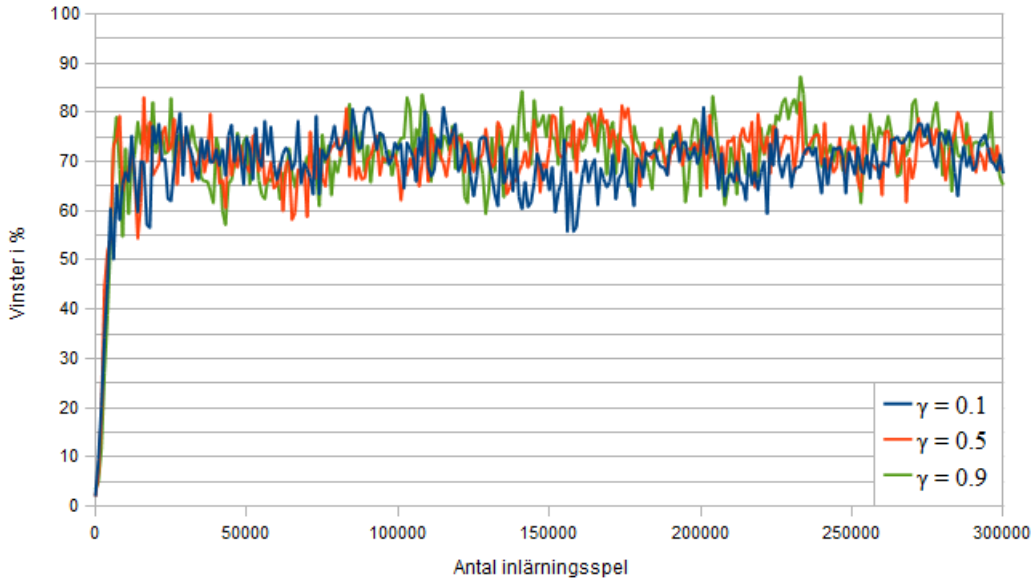
Figur 4.1. Graf över $\alpha = 0.1$

I figur 4.1 ser man att för de olika värdena på γ är inlärningshastigheten inledningsvis relativt lika. Senare in i inlärningsfasen ser man dock att agenten med $\gamma = 0.9$ får en högre vinstprocent och har därmed lärt sig bättre än de agenter med lägre γ. Man kan även se tendenser till att $\gamma = 0.9$ stabiliserar sig i slutet av samtliga spelomgångar, till skillnad från de mindre värdena.



Figur 4.2. Graf över $\alpha = 0.5$

I figur 4.2 ser man att vinstprocenten varierar kraftigt beroende på om γ är stort, 0.9, eller litet, 0.1. För $\gamma = 0.5$ blir kurvan ungefär likadan som för 0.1. Precis som figur 4.1 är inlärningshastigheten väldigt lik inledningsvis för att sedan variera beroende på vilket värde γ är satt till.



Figur 4.3. Graf över $\alpha = 0.9$

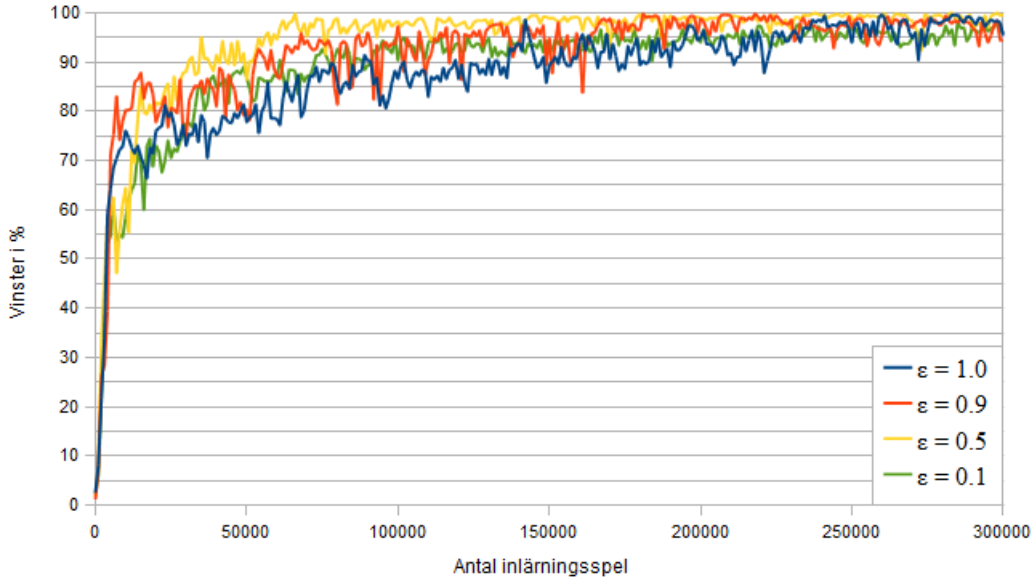
I figur 4.3 ser man att när α är lika med 0.9 och därmed väldigt högt minskar effekten av γ. Inlärningen stabiliseras aldrig och inlärningskvaliteten blir inte lika bra som för lägre värden på α.

Bästa kombinationen av alpha och gamma

För att avgöra vilken kombination av α och γ som är bäst för inlärning jämförs de tre diagrammen. I figur 4.3 ser vi att för $\alpha = 0.9$ ligger vinstprocenten runt 70 vinstprocent i snitt. Samtliga grafer i figuren visar också påtaglig instabilitet. Detta gör att $\alpha = 0.9$ inte lämpar sig för optimal inlärning. För $\alpha = 0.5$ ser vi fortfarande instabilitet oavsett värdet på γ. Dock ger $\gamma = 0.9$ betydligt högre vinstprocent totalt. I figur 4.1 där $\alpha = 0.1$ visar de tre graferna på stabilitet. Däremot når inte alla lika hög vinstprocent. $\gamma = 0.9$ är den enda av alla 9 grafer som når stabilt över 90 vinstprocent. Den initiala inlärningen, för de 25000 första inlärningsspelen, är likvärdig för de nio graferna. Det bästa värdet blir därför 0.1 på parametern α och 0.9 på parametern γ. Vi har därför valt att använda dessa värden i kommande analyser i rapporten.

4.1.2 Epsilon

För att bestämma det bästa värdet, vid inlärning, på ϵ användes i princip samma testmetod som för de tre föregående testerna. Skillnaden nu är att α och γ är tilldelade de bästa värdena vi fann tidigare. ϵ testas för värdena 0.1, 0.5 och 0.9. Den bästa körningen från tidigare test, med $\epsilon = 1.0$, finns med som referens.



Figur 4.4. Graf över $\alpha = 0.1$, $\gamma = 0.9$ och $\epsilon = 0$, 0.1, 0.5, 0.9 och 1

För samtliga testade värden når agenten över 90 vinstprocent. Skillnaden mellan graferna är hur snabbt de når den gränsen. $\epsilon = 0.5$ är det värde som snabbast når dit, även om agenten med de andra tre värdena också når den gränsen relativt snabbt. Ser man till stabilitet är $\epsilon = 0.5$ den som snabbast når en stabil vinstprocent över 90.

Undersökningen visar därför att $\epsilon = 0.5$, $\alpha = 0.1$ och $\gamma = 0.9$ är dem bästa inlärningsparametrarna.

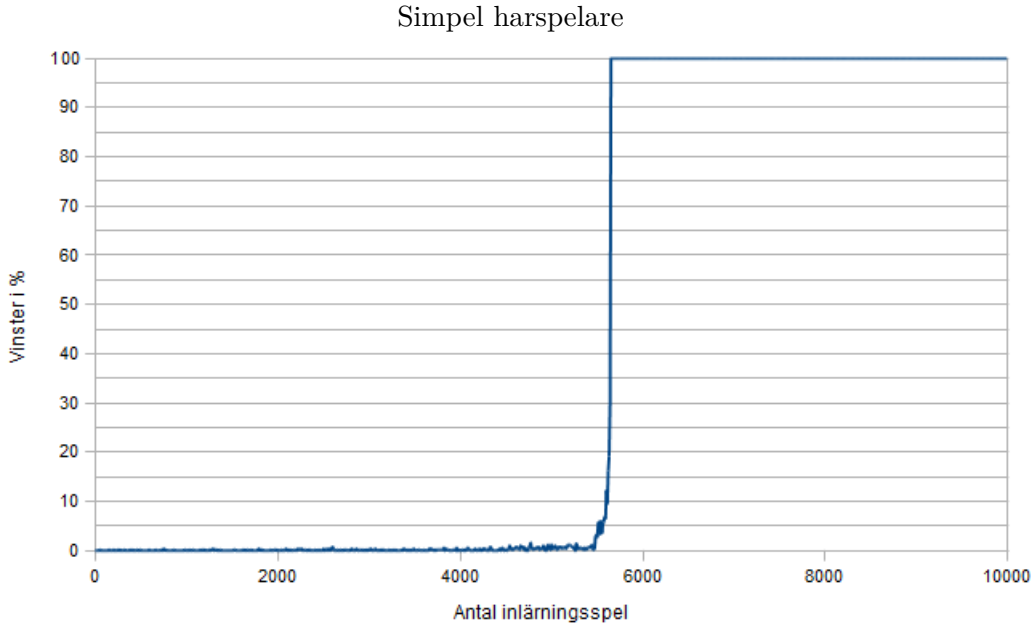
4.2 Inlärning mot en simpel strategi

Hittills har Q-learning agenten bara spelat mot den slumpmässiga harspelaren. Därför tränas agenten mot en spelare som spelar på samma sätt varje gång. Detta för att se om agenten kan lära sig att vinna mot en specifik, men dock simpel, strategi. Testet utförs på följande sätt:

1. Först körs 10 inlärningsspel med $\alpha = 0.1$, $\gamma = 0.9$ och $\epsilon = 0.5$.

2. Sedan pausas inlärningen och 500 spelomgångar med inlärningsparametrar satta till 0, dvs agenten lär sig inget nytt och använder sig endast av tidigare inlärda kunskaper, körs En vinstprocent av de 500 spelomgångarna räknas ut och visas i grafen.

3. Efter 500 spelomgångar med ovan nämnda inställningar återupptas agentens inlärning. Värdena som visas i grafen är vinstprocenten för vart 10:e inlärningsspel av sammanlagt 10 000 spelomgångar.



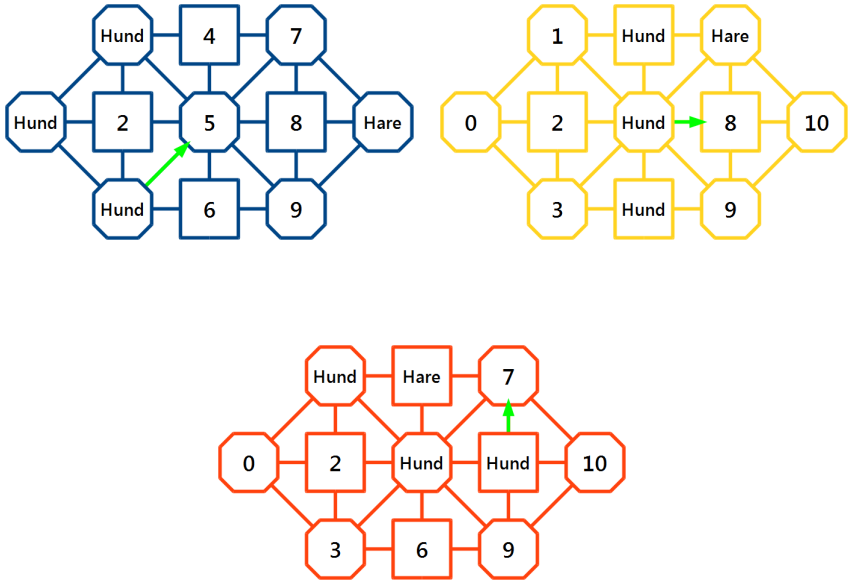
Figur 4.5. Graf över att Q-learning agenten lär sig den simpla harspelarens strategi.

Grafen visar att agenten lär sig att vinna mot den simpla harspelaren. När agenten har hittat en vinnande motstrategi lär den sig väldigt snabbt att det är på exakt det viset den alltid kan vinna. Det tar ungefär 5000 inlärningsspel för agenten att lära sig.

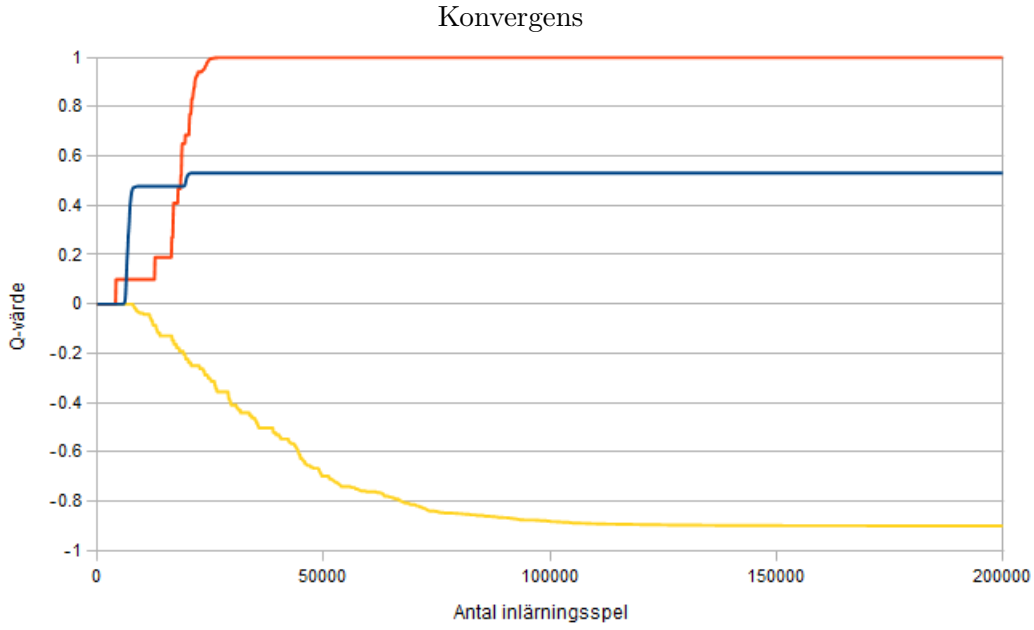
4.3 Konvergens

Tidigare test har visat att agenten kan lära sig att vinna. För att ta reda på huruvida algoritmen konvergerar för något tillstånd testas tre tillstånd och handlingar för att undersöka om dess Q-värden når ett slutgiltigt värde. I figur 4.6 visas de tillstånd vars Q-värden konvergenstestas. De gröna pilarna anger vilken hund som ska gå till den utpekade spelrutan, alltså vilken handling som ska utföras.

1. Först körs 100 inlärningsspel med $\alpha = 0.1$, $\gamma = 0.9$ och $\epsilon = 0.5$.
2. Sedan avläses Q-värdet för det aktuella tillståndet och den aktuella handlingen.
3. Detta upprepas i sammanlagt 200 000 inlärningsspel. I grafen visas sedan hur Q-värdet ändras beroende på hur många inlärningsspel som har körts.



Figur 4.6. Tillstånden och handlingarna för vars Q-värden konvergenstestas



Figur 4.7. Graf över att Q-värdena konvergerar.

Graferna visar att alla de Q-värden vi testade konvergerar. Tillståndet ritat i rött har en handling som ger vinst. Följaktligen konvergerar dess Q-värde mot 1, som alltså är belöningen för vinst. Att ha en hund på ruta 5 torde konvergera mot ett positivt värde då ruta 5 är den rutan som blockerar flest flyktvägar för haren. Det blåa tillståndets graf visar att handlingen att ställa en hund på ruta 5 faktiskt också konvergerar mot ett positivt värde. Tillståndet markerat med gult har en handling som antagligen leder till att haren flyr. Eftersom förlust bestraffas med -1.0 konvergerar det Q-värdet tillslut mot ett negativt värde.

Kapitel 5

Diskussion

Här diskuteras resultaten som presenterats i kapitel 4.

5.1 Optimering av inlärningsparametrar

Det man kan se genom alla testar vid låga α, dock inte lika med noll, är att inlärningen blir mer stabil. Enligt oss beror det på, eftersom α är inlärningshastigheten, att vid låga värden kommer agenten endast uppdatera sina Q-värden med små värden. Detta kan man utläsa ur formel 3.1. Det leder till att ett Q-värde som under lång tid har ökat och visat sig fördelaktigt inte kan förändras åt motsatt håll på några få spelomgångar. För ett högt α kan ett Q-värde som visat sig fördelaktigt förändras åt det motsatta på en enda spelomgång, vilket orsakar instabilitet hos agenter med ett högt α.

För att vinna på Hare and Hounds krävs det att man har en strategi som sträcker sig ett antal drag fram, och inte endast beaktar nuet. Detta uppnås genom att tilldela agenten ett högt γ, vilket våra undersökningar visar.

Det bästa värdet på ϵ som erhölls var 0.5. Det visar att det är lika viktigt att befästa sina redan inlärda policys som att slumpmässigt hitta nya policys. Vad vi menar är att medan en agent lär sig kan den använda redan inlärda handlingar och därmed minskar den risken för att slumpmässigt välja sämre handlingar.

5.2 Inlärning mot en simpel strategi

Q-learning är effektiv på att lära sig att vinna mot en specifik strategi. Om agenten lyckas vinna endast några få spelomgångar räcker det för att ha lärt sig att spela på samma sätt varje spel. För simpla strategier, som vår, leder detta till vinst i 100% av fallen.

5.3 Konvergens

Om hela Q-tabellen har konvergerat kommer agenten att ha minst en handling som är den optimala i varje tillstånd, och den kan då inte lära sig något mer. I figur 4.7 visas att tre Q-värden har konvergerat, med säkerhet, efter 150 000 inlärningsspel. Eftersom Q-learning bevisligen konvergerar antar vi att Q-tabellen har konvergerat efter 150 000 inlärningsspel mot den slumpmässiga harspelaren. Detta påvisar att vår Q-learning hundspelare lär sig att spela Hare and Hounds med en ultimat strategi, oavsett harspelarens strategi.

Kapitel 6

Slutsats

Här drar vi slutsatser från kapitel 5.

Kan en agent som använder sig utav Q-learning lära sig att spela strategibrädspelet Hare and Hounds?

En agent som använder sig utav Q-learning kan utan tvekan lära sig att spela Hare and Hounds. Samtliga undersökningar i denna rapport visar på detta.

Detta går även att implementera på alla andra brädspel. Begränsningen i detta fall är tillståndsrymden som för vissa spel kan bli allt för stor. Anledningen till att det går så bra att implementera på brädspel är att man endast behöver definiera belöningen, vilket inte är så komplicerat. Det räcker med att ge belöning för vinst, lika eller förlust.

Vilka värden på parametrarna epsilon, alpha och gamma är de optimala vid inlärning?

Våra experiment visar att den bästa kombinationen på inlärningsparametrarna är $\epsilon = 0.5$, $\alpha = 0.1$ och $\gamma = 0.9$. Detta gäller framförallt för att lära sig spela Hare and Hounds men liknande värden borde även gälla för andra brädspel.

Kommer agenten, efter inlärning, att klara sig mot en mänsklig motståndare?

Agenten kan absolut lära sig att spela mot mänskliga spelare. När vi har lärt upp agenten tillräckligt länge har vi själva ingen chans att slå den. Strategin den lär sig tycks vara näst intill omöjlig att slå. Men bara för att agenten lär sig spelet så otroligt bra betyder det inte nödvändigtvis att spelet blir roligt att spela, då man förlorar hela tiden.

Litteraturförteckning

[1] E. Alpaydin. *Introduction to machine learning*. The MIT Press, 2004.

[2] Martin Gardner. Mathematical games. *Scientific American*, October 1963.

[3] Imran Ghory. Reinforcement learning in board games. Teknisk rapport CSTR-04-004, Department of Computer Science, University of Bristol, May 2004.

[4] S.J. Russell och P. Norvig. *Artificial intelligence: a modern approach*. Prentice hall, third edition utgåvan, 2010.

[5] R.S. Sutton och A.G. Barto. *Reinforcement learning: An introduction*. The MIT press, 1998.

[6] C.J.C.H. Watkins och P. Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[7] Wikipedia. Hare and hounds, 2011-04-12. `http://en.wikipedia.org/wiki/Hare_and_Hounds_(board_game)`.

[8] Wikipedia. Hare games, 2011-04-12. `http://en.wikipedia.org/wiki/Hare_games`.