13.10.2024

Eirik Tennøfjord

Student

Et bilde som inneholder skjermbilde, tekst, programvare, Fargerikt

Automatisk generert beskrivelse

Q-Learning

Karaktersatt oppgave 1 DTE2602 – Introduksjon Maskinlæring og AI

Innhold

[1. Terminologi 2](#_Toc179498800)

[2. Introduksjon 2](#_Toc179498801)

[3. Teori 3](#_Toc179498802)

[4. Metode 3](#_Toc179498803)

[4.1 - Monte Carlo 4](#_Toc179498804)

[4.2. Q-Learning/Greedy 5](#_Toc179498805)

[4.3. Epsilon-Greedy 5](#_Toc179498806)

[4.4. Diverse 5](#_Toc179498807)

[5. Resultat 6](#_Toc179498808)

[6. Diskusjon og konklusjon 6](#_Toc179498809)

[7. Kilder 7](#_Toc179498810)

# Terminologi

Agent

Miljø

Q-Learning

Monte Carlo

Epsilon-Greedy

Belønning

Q-Matrise

Episoder/Epochs

|  |  |
| --- | --- |
| Agent | En agent er selve koden. Eller den som tar beslutninger. Denne bruker informasjon fra miljøet for å utføre handlinger for å oppnå et mål. |
| Miljø | Miljøet er selve omgivelsene agenten opererer i. Dette kan være et fysisk miljø, eller en simulert verden. Miljøet gir tilbakemelding til agenten i form av belønning/straff og tilstand. |
| Episoder (Epochs) | En episode er en hel sekvens fra start tilstand, til slutt tilstand. For å trene agenten er det nødvendig å ha flere episoder, slik at man gradvis kan forbedre seg. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Introduksjon

I dagens samfunn har utviklingen innenfor kunstig intelligens (KI) skutt fart. Eksempler på dette kan være chat bot, selvkjørende biler, selvkjørende ferger/båter og sorteringsroboter. Flere av disse punktene dreier seg om navigasjon, som er en sentral del av KI. Navigasjon er en av de grunnleggende utfordringene innenfor robotikk og maskinlæring. Det å kunne navigere i et miljø, for deretter å identifisere hindringer og finne den mest effektive veien kan være avgjørende for flere felt.

En viktig del av det å kunne navigere i et miljø, er forsterkningslæring, der vi bruker en kode (agenten) til å navigere og ta beslutninger basert på miljøet. Q-learning og Monte Carlo er to kjente metoder innenfor forsterkningslæring. Begge disse metodene lærer ved å samle opp belønning over tid. Q-learning lærer ved å kontinuerlig oppdatere en Q-matrise etter hvert som miljøet blir utforsket. Monte Carlo summerer opp belønning og sammenligner disse for å finne den mest effektive ruten.

En annen strategi for å navigere er en metode vi kaller epsilon-greedy. Denne kombinerer q-learning, og monte carlo. Ved å benytte denne metoden, så vil man bruke kjent informasjon om hvilke ruter som gir høyest belønning, samtidig som man vil kunne oppdage potensielt bedre ruter, ved å gjøre tilfeldige handlinger.

For å studere disse forskjellige læringsmetodene er et 2D-kart et egnet miljø. Gjennom bruk av belønningsmatriser og simuleringer kan agenten lære seg å finne effektive ruter fra start punkt til slutt punkt, samtidig som agenten kan manøvrere seg rundt hindringer.

Denne rapporten vil undersøke hvordan man med ulike navigasjonsstrategier, som Monte Carlo og Q-learning kan finne de mest effektive rutene i et 2D-kart. Formålet er kunne vurdere hvordan de ulike metodene fungerer, samt hvor effektive de er. Det er også ønskelig å se på antall episoder (epochs), og hvordan dette spiller inn på de forskjellige strategiene. I tillegg til dette skal det også være forskjellig type landskap som fjell og vann, noe som kan skape problemer når man skal bevege seg innenfor disse områdene. Vi tar utgangspunkt i at det er en robot som skal utforske ukjent terreng.

Hypotesen i denne rapporten er at en kombinasjon av disse metodene kan gi en optimal løsning for robotens navigasjon. Q-learning vil være mest effektiv hvor man har et stort antall episoder, mens epsilon-greedy kan gi bedre resultater med et mindre antall episoder. Gjennom simulering vil rapporten vise hvordan roboten lærer og forbedrer sine valg over tid, og hvordan den finner de mest effektive veiene fra start til slutt.

#### MÅ PRESENTERE OPPGAVEN PÅ EN MÅTE #####

# Teori

## 3.1 - Monte Carlo Simulering

Monte Carlo har fått navnet sitt etter kasinoet i Monaco, som er et kjent kasino.

## 3.2 - Markov Decision Process (MDP)

Teori her…

Et bilde som inneholder tekst, diagram, sirkel, ball

Automatisk generert beskrivelse  
Figur 2.0 – Q-læring

Teori om q-læring

## 3.3 - Bellman likningen

-------------------------------------------------------------------------------------------------

* Referanse og forklaring til bellmanlikning
* Markov-antakelsen
* Monte Carlo simulering
* Minst én relevant referanse til relevant problematikk knytt til navigasjon

"Teori-delen beskriver teoretiske konsepter som er relevante for rapporten, for eksempel algoritmer, matematiske ligninger, database-modeller, osv. Husk at elementene som brukes må beskrives i teksten (f.eks. hva hver enkelt variabel i en ligning betyr). Husk også referanser."

----------------------------------------------------------------------------

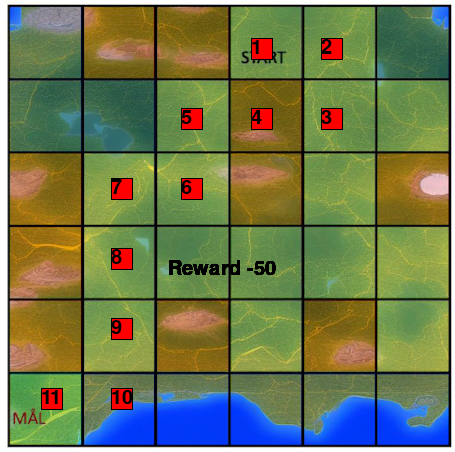
# Metode

For å utvikle agenten til å navigere i 2D-kartet, ble de benyttet forskjellige metoder innenfor forsterkningslæring. Dette inkluderte: Monte Carlo, Q-learning og Epsilon-Greedy policy. Selve arbeidet ble gjennomført i Python, og ved hjelp av visualiseringsverktøyet "PyGame".

Et bilde som inneholder tekst, skjermbilde, Font, line

Automatisk generert beskrivelse  
*Figur 3.1: Her kan man velge hvilken metode som skal benyttes.*

I denne rapporten er det laget klart for tre forskjellige typer maskinlæringsmetoder. Disse kan man velge ved hjelp av «radiobuttons» i programmet (figur 3.1). Her kan man også skrive inn hvor mange episoder som skal benyttes for de forskjellige metodene.

  
*Figur 3.2: Visning av beste sti*

Når koden er ferdig kjørt, så vil man få opp i GUI hvilken sti programmet har kommet frem til er den beste.

## 4.1 - Monte Carlo

Monte Carlo ble iverksatt ved at man setter agenten/roboten i start posisjon, deretter fikk den gjøre tilfeldige handlinger som i dette tilfellet var opp, ned, venstre og høyre. Hver gang man kommer til en ny tilstand vil man oppdatere episodens totale belønning.

Et bilde som inneholder skjermbilde, Fargerikt, kvadrat

Automatisk generert beskrivelse  
*Figur 3.3: Agent gjør en handling som vil føre man utenfor kartet.*

I Monte Carlo, kan man også gjøre valg som fører til at man kan forsøke å gå gjennom ytterkanten/veggen på kartet (som vist i figur 1). Denne gir ifølge belønningsmatrisen en negativ belønning. I denne rapporten er det valgt å ikke summere opp straffen fra å prøve å gå utenfor kartet, siden agenten uansett ender i samme tilstand. Dette gjelder da kun for Monte Carlo

## 4.2. Q-Learning/Greedy

Q-Læringen ble implementert ved å initialere en Q-matrise som holdt styr på de forskjellige belønningene for hver tilstand og handling. Agenten startet fra en tilfeldig posisjon i kartet og utfører en handling basert på valgt policy (ϖ).

For hvert steg vil belønningsmatrisen bli oppdatert i henhold til Bellman ligningen. Etter at man har kjørt alle episodene, så vil man kalle funksjon for å finne sti basert på verdiene i q-matrisen. For å finne beste sti basert på q-matrisen har det blitt iverksatt en greedy funksjon, som i en gitt tilstand skal velge den handlingen som gir høyest mulig belønning.

## 4.3. Epsilon-Greedy

Denne metoden bygger på de samme prinsippene som q-læring og Monte Carlo, og kombinerer disse. Ved hjelp av en parameter satt i robot filen (epsilon), vil man kunne justere på hvor ofte man skal gjøre en tilfeldig handling, i stedet for å gå etter den som gir mest belønning. Dersom man har en høy verdi på epsilon (mellom 0 og 1), så vil man oftere gjøre tilfeldige handlinger.

## 4.4. Diverse

For greedy explotation funksjonen som blir brukt i både Epsilon og Q-learning i denne rapporten, så er det også gjennomført en funksjon for «visited». Dette er for å forhindre at man går i en uendelig loop mellom to ruter. Dersom alle nabo-tilstandene er besøkt, så vil man velge en tilfeldig.

Visited funksjonen er en matrise med alle tilstandene. Hver av tilstandene har en påfølgende verdi som er True/False. Det er også lagt til et ekstra ledd i hver kant for å lagre veggene som «visited». Disse blir satt til besøkt fra start.

Forsøkene i denne rapporten er utført med følgende verider:

|  |  |  |
| --- | --- | --- |
| Verdi | Variabel | Forklaring |
| 0.4 | Alpha (Læringsrate) | Denne er satt til en lavere verdi siden vi opererer i et statisk miljø. Det er ikke noen store endringer. Dersom det hadde vært et mer dynamisk miljø, ville det vært mer nyttig med en høy læringsrate. [1] |
| 0.9 | Gamma (Discount factor) | Denne er satt til en høy verdi, fordi belønningen kommer senere. Når denne er høy, vil man derfor vektlegge fremtidig belønning. Dersom man var usikker på belønningen kunne man brukt en lavere verdi her. [1] |
| 0.1 | Epsilon | Denne er satt til denne verdien etter en del prøving og feiling. Dersom man hadde hatt mer tid, skulle man helst hatt iverksatt en funksjon som gradvis justerte ned epsilon variabelen utover i episodene. |
| 0 | Hvitt felt | Har valgt å gi denne verdien «0», siden man ikke skal bli straffet for å gå på disse feltene, men samtidig kommer belønningen senere. |
| -50 | Rødt felt | De røde feltene (fjellområdene) har fått verdien «-50». Dette fordi man skal helst unngå å gå på disse feltene, med mindre man må. |
| -100 | Blått felt | De blå feltene har fått verdien «-100». Dette fordi at det fortsatt skal være mulig å gå innom disse feltene, men man bør heller prioritere hvitt eller rødt fremfor blått. |
| -1000 | Vegger | For å unngå at agenten prøver å gå inn i veggene, så har disse fått en høy straff. |

* Alle valg skal være godt begrunnet og dokumentert med referanse til koden.
* Hele arbeidsprosessen er sammenhengende og dokumentert, ikke unødvendig delt opp.
* Fremgangsmåten er klart og tydelig beskrevet slik at andre kan gjenta ekspermentene.

"Beskrivelse av hvordan arbeidet ble gjennomført i praksis - hvilke verktøy og metoder som ble brukt, implementasjonsdetaljer, osv. Bruk et objektivt språk som fokuserer på hva som ble gjort, og unngå &amp;amp;amp;quot;dagbok-format&amp;amp;amp;quot;. Dokumenter valg som er tatt i koden, og inkluder gjerne kodesnutter der det er relevant, men unngå å liste opp all kode (som skjermbilder eller tekst)."

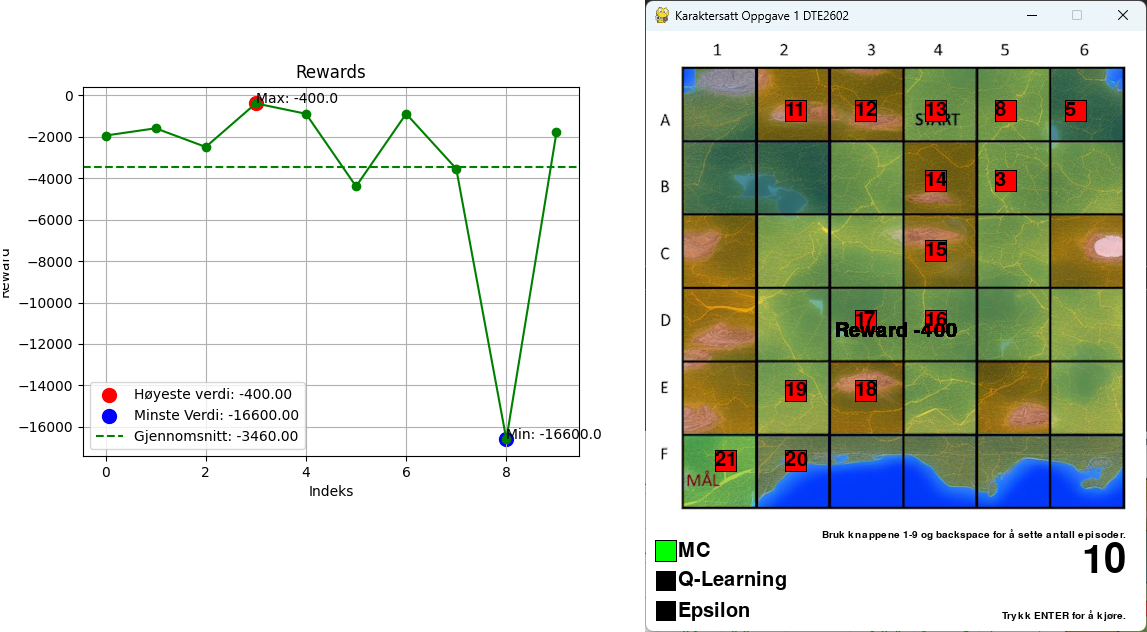
# Resultat

Resultat her

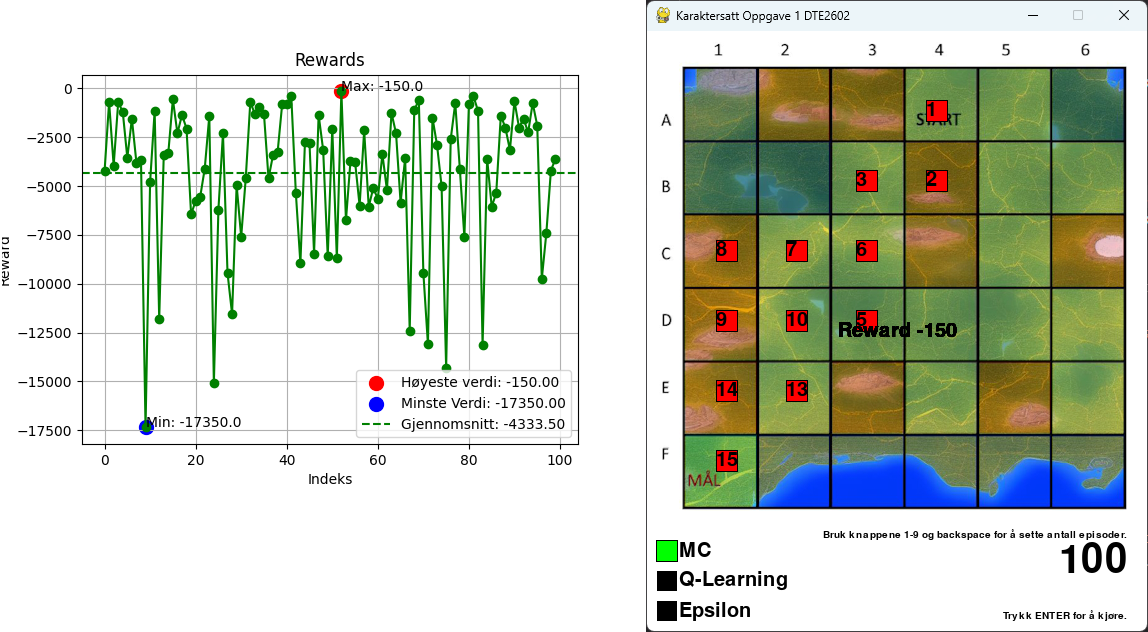
I denne delen av rapporten er det bestemt å vise resultatet av de forskjellige typene læring som er gått gjennom tidligere.

## 5.1 Monte Carlo

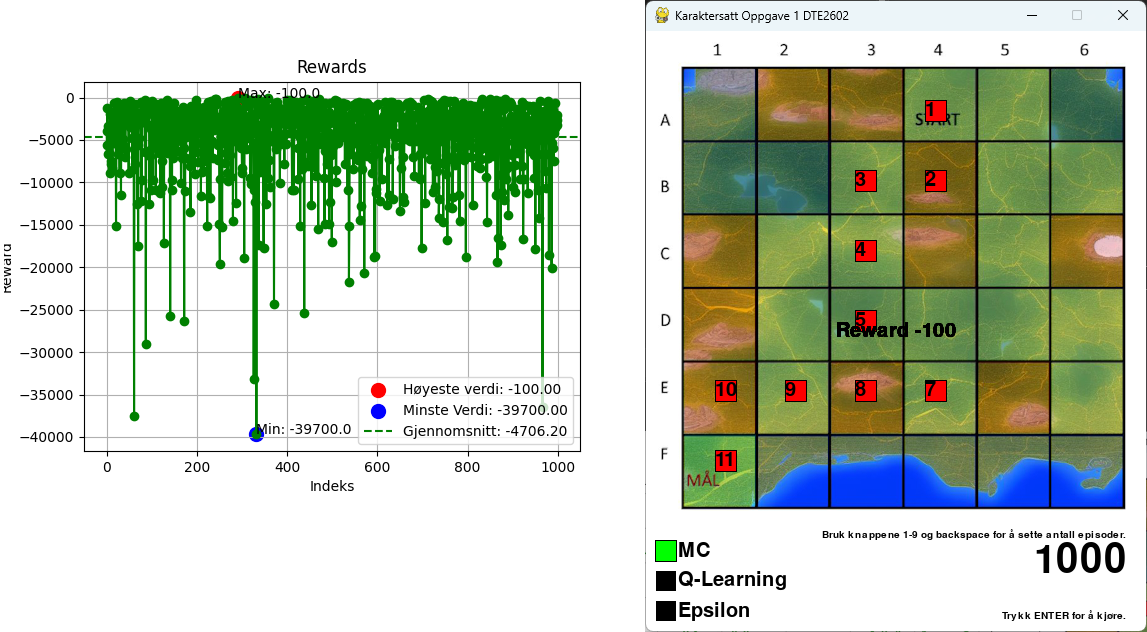
For å komme frem til et resultat for denne metoden har jeg valgt å vise resultatet av simulering kjørt med 10, 100 og 1000 episoder.

  
*Figur 5.1.1: Monte Carlo med 10 simuleringer*

Når det gjelder simuleringen med 10 episoder, så er det svært mye tilfeldigheter knyttet til dette resultatet. I denne spesifikke testen var den beste oppnådde belønningen -400, og den dårligste belønningen var -16,600. Noe som er et svært stort sprik.

  
*Figur 5.1.2: Monte Carlo med 100 simuleringer*

Når vi økte antallet episoder til 100, så ser vi at en bedre rute er funnet. Den er ikke så veldig mye bedre enn ruten som ble funnet med 10 episoder. Samtidig ser vi at den dårligste belønningen oppnådd er -17,350. Og gjennomsnittet er -4333.5.

  
*Figur 5.1.3: Monte Carlo med 1000 simuleringer*

Til slutt har det blitt gjennomført 1000 episoder. Med denne simuleringen har vi funnet den til nå beste ruten, men det er fortsatt ikke den optimale ruten. Samtidig kan vi bemerke oss at belønningen i dette spesifikke forsøket kun har en forskjell på 50 mellom 100 og 1000 episoder.

## 5.2 Q-Læring/Greedy

For testing i henhold til q-læring har det blitt bestemt å gjøre testene litt annerledes enn på Monte Carlo, på grunn av deres virkemåte. De følgende bildene vil vise den beste stien etter 10, 25 og 50 episoder. For hver av disse er det kjørt 100 simuleringer, for å få grafer vi kan analysere. Her begynner man også å få den optimale ruten, så i stedet for å vise grafer som på Monte Carlo, vil det her bli vist søyle diagram med antall optimal rute og ikke optimal rute (Optimal rute vil si den beste ruten oppnåelig).

Et bilde som inneholder diagram, skjermbilde, line, Rektangel

Automatisk generert beskrivelse  
*Figur 5.2: Q-læring med «Greedy route» for 10, 25 og 50 episoder med 100 simuleringer.*

Etter 100 simuleringer med 10 episoder, ser vi at vi begynner å få noen simuleringer som finner den optimale ruten. Men det er fortsatt en god del som er feil.

Når vi øker antallet episoder til 25, ser vi at resultatet begynner å bli mye bedre. I denne spesifikke testen ble det 39% av simuleringene som klarte å finne den mest optimale stien.

I den siste målingen med 50 episoder, kan vi se at andelen med optimal rute har blitt mye større enn andelen som ikke klarer å finne den. Men ut ifra disse resultatene kan vi se at det fortsatt kan være en del feilmargin.

## 5.3 Epsilon-Greedy

Den siste metoden er epsilon greedy, denne har det blitt bestemt å teste på samme måte som q-læring/greedy.

Et bilde som inneholder diagram, line, skjermbilde, Rektangel

Automatisk generert beskrivelse  
*Figur 5.3: Epsilon med «Greedy route» for 10, 25 og 50 episoder med 100 simuleringer*

Resultatet for denne testen er svært lik resultatet i q-learning/greedy. Har var det forventet at epsilon skulle være mer effektiv ved mindre episoder.

------------------------------------------------

Resultater som er relevant til problemstillingen er presentert objektivt og oversiktelig

"Presentasjon av resultatene av arbeidet. Formidles gjerne i form av tabeller, grafer, illustrasjoner og/eller bilder i tillegg til tekst. God struktur er viktig for at leseren skal skjønne sammenhengen mellom resultatene og tidligere beskrevet teori og metode. Kommenter gjerne kort i teksten for å fremheve de mest interessante / overraskende resultatene, men pass på å ikke bli for subjektiv."

# Diskusjon og konklusjon

Diskusjon og konklusjon

Dersom man har flere verdier i q-matrisen som er den samme (som når man initialiserer den), så vil man i denne epsilon-greedy funksjonen alltid velge «den første» av de beste valgene.

  
*Figur 5.1 – Hvordan metoden for epsilon-greedy er implementert.*

Ble brukt for lang tid på GUI, hadde vært like greit med en meny ved hjelp av kommandovinduet.

I Monte Carlo funksjonen er det ikke laget til med en «visited» matrise, dette burde også implementeres her.

Burde lage en funksjon for å gradvis endre epsilon verdien. Mye usikkert i starten. Kunne heller hatt høy random de første episodene, og deretter gradvis justert ned epsilon.

Det er ikke lagt til konvergering i dette programmet. Både for epsilon-greedy og q-læringen, så man må alltid fullføre alle episodene selv om det ikke er noen endringer. Dette ble ikke prioritert i dette tilfellet.

------------------------------------------------

* God selvinnsikt som står til levert kode og oppnådde resultater
* Drøfting av mulige feilkilder, satt i kontekst med framtidig arbeid.

"Diskusjon rundt resultatene: Hvilke resultater er mest interessante? Er resultatene som forventet, eller ikke? Prøv å sette oppnådde resultater i sammenheng med opprinnelig problembeskrivelse/introduksjon. Konklusjonen skal kunne leses ganske frittstående, uten å kjenne alle detaljene i rapporten. Dersom det er aktuelt å fortsette arbeidet, kommenter gjerne hva som bør gjøres videre."

------------------------------------------------

# Kilder

Figur 1 – Utklipp av programmet, med pil for at man prøver å gå gjennom vegg.

[1] – Forelesning om hyperparametere

[2] - https://www.techtarget.com/searchenterpriseai/definition/Q-learning