Q learning rapport

Contents

[Introduksjon 1](#_Toc149784017)

[Metode 1](#_Toc149784018)

[Hva er monte carlo exploration 1](#_Toc149784019)

[Hva er q learning 1](#_Toc149784020)

[Resultater 2](#_Toc149784021)

[Monte carlo resultater 2](#_Toc149784022)

[Q learning resultater 4](#_Toc149784023)

# Introduksjon

Her lager vi en robot, som prøver å finne korteste vei fra start til slutt, ved bruk av monte carlo exploration eller q learning. Selve problemstillingen er å finne beste mulige vei med mest mulig poeng. Vi viser eksempler fra begge metodene; starter først med monte carlo exploration deretter q learning, viser også fordeler og ulemper ved metodene.

# Metode

Vi bruker et empirisk forsøk og prøver å lage eksperimenter også tar vare på resultater.

# Hva er monte carlo exploration

Det er en måte å utforske et miljø på ved å bruke randomiserte valg, agenten har ingen anelse hvor nært målet han er. Metoden bruker også en reward matrise for å gi belønning til agenten for å gi en indikasjon på hvor bra valg det er. Metoden trenger veldig mange forsøk for å finne absolute beste vei, med best mulig straff. Den tar vare på stien og total belønning for en simulasjon og lagrer dette i en liste, til slutt så retunerer den beste stien med maksimal belønning.

# Hva er q learning

Q learning er en model fri forsterkning læring algoritme og bruker en q matrise for å velge beste handling; den lærer basert på belønning matrisen ved bruk av bellman funksjonen.

Det er likt monte carlo exploration ved at den utforsker miljøet og kan ta tilfeldige valg og har en reward matrise. Fordelen med q learning er q verdi (quality) som gir en indikasjon på fremtidige belønninger vis man tar et valg. Man bruker en q matrise for å lagre disse verdiene. Med q learning kan man finne mest optimale vei ifølge en gitt reward matrise. For å regne ut q verdier bruker man et matematisk funksjon som er bellman funksjonen. der α er learning rate og γ er discount factor; s er state og a er action, s’ er neste state og a’ er alle handlinger.

Metoden klarer å finne beste vei mye raskere en monte carlo exploration siden den utvikler en sans for miljøet ved trening. Ved å justere på hyper parametene som alpha og gamma kan man finne kortere vei mye raskere.

# Resultater

Følgende resultater bruker numpy.random.seed(0).

## Monte carlo resultater

Belønning matrisen er:

[-100, -50, -50, -1, -1, -100],

[-100, -100, -1, -50, -1, -1],

[-50, -1, -1, -50, -1, -50],

[-50, -1, -1, -1, -1, -1],

[-50, -1, -50, -1, -50, -1],

[200, -100, -100, -100, -100, -100]

V har bestemt at følgende reward matrise best svarer på problemstillingen, på mest effektivt måte.

Med tilsvarende -1000 for å treffe en vegg.

-100 for å gå i vann.

-50 for å gå i fjell.

-1 for vanlige terreng

200 for mål.

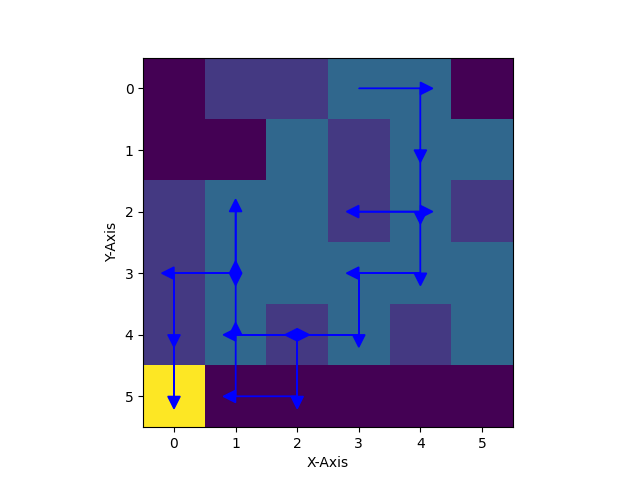
Har bestemt -1 for vanlige terreng for at agenten ikke skal utnytte vanlige terreng, og ikke gå gjennom som mange som kan uten straff.

Belønningene er balansert i forhold til hverandre, sånn at roboten skal velge mest effektive vei.

Følgende figurer er resultat av monte\_carlo\_exploration(simulations, start\_state, goal\_state) funksjonen.

Med start state = (0,3) og goal\_state = (5,0).

Monte carlo exploration med 100 simuleringer



Monte carlo exploration med 100 simuleringe. Fra figuren kan man se at den ikke klarte å finne beste vei, fra bare 100 simuleringer; man trenger mere simuleringer for å få bedre resultat.

Monte carlo exploration med 100 000 simuleringer

A graph with arrows pointing to a graph

Description automatically generated with medium confidence

Monte carlo exploration med 100 000 simulering. Som man kan se, kan forsatt ikke finne optimale stien siden den valgte å gå gjennom fjellet i (1,3), i stedet for å gå rundt.

Monte carlo exploration med 1 000 000 simuleringer

A graph of a graph

Description automatically generated with medium confidence

Vis man kjører dette 1 000 000 ganger klarer den til slutt å finne best vei, men det krever mye tid og resurs og er kost krevende.

## Q learning resultater

Følgende belønning matrise er brukt for q learning forsøkene.

REWARD\_MATRIX = np.array([

[-200, -100, -100, -1, -1, -200],

[-200, -200, -1, -100, -1, -1],

[-100, -1, -1, -100, -1, -100],

[-100, -1, -1, -1, -1, -1],

[-100, -1, -100, -1, -100, -1],

[200, -200, -200, -200, -200, -200]

])

Følgende q matrise er brukt 6\*6\*4, der for hver rad og for hver kolone er det 4 handlinger, q matrisen lagrer altså q verdiene for 4 handlinger for hver tilstand som er 36.

Alpha = 0.1

Gamma = 0.9

Epsilon = 0.8

I denne rapporten har vi bestemt å bare bruke konstant epsilon, for å få mindre feil i utforskingen.

A grid map with arrows and lines

Description automatically generated

Q learning med 1000 episoder

Her kan man se beste handlinger for hver state. Den klarte å finne beste vei mye fortere sammenlignet med monte carlo exploration.

A grid map with arrows and lines

Description automatically generated

Samme forsøket men med bare 100 episoder

Det virker som agenten trenger ikke mere enn 100 episoder for å finne optimale vei, med gitt hyperparametere.

Vis man prøver å kjøre med for eksempel 70 episoder, for dette eksemplet; regner den ut q verdier, men det kan hende det oppstår løkker i tabelen fordi den ikke har utforsket nok.

A grid map with red and black squares

Description automatically generated

Q learning med 70 episoder

Som man kan se, så er det løkker i handlings kartet, spesielt i (0,3) til (0,4) der den vil gå fram å tilbake. Resten har den klart å finne fram ganske bra.

A grid map with red and black squares

Description automatically generated

Q learning med 10 episoder.

Med 10 episoder, har den ikke utforsket nok og det finnes mange løkker i matrisen.

Hva skjer vis man kjører 10000 episoder?

A grid map with arrows and lines

Description automatically generated

Q learning med 10 000 episoder

Her kan man se ingen stor forskjell fra 1000 episoder eller 100 episoder.

Siden det har skjedd liten forandring fra bare 100 episoder, kan man si da at algoritmen har konvergert og det har ingen hensikt å kjøre flere episoder.

Det finnes mange metoder for å finne konvergens på, i dette tilfelle kan man bare telle uforandringer, vis det er 10 eller mere, kan man si det er konvergert. Vi prøvde å summere forskjellene, men klarte ikke å benytte dette i denne sammenhengen.