TEKNOLOGISK
INSTITUT

Rapport

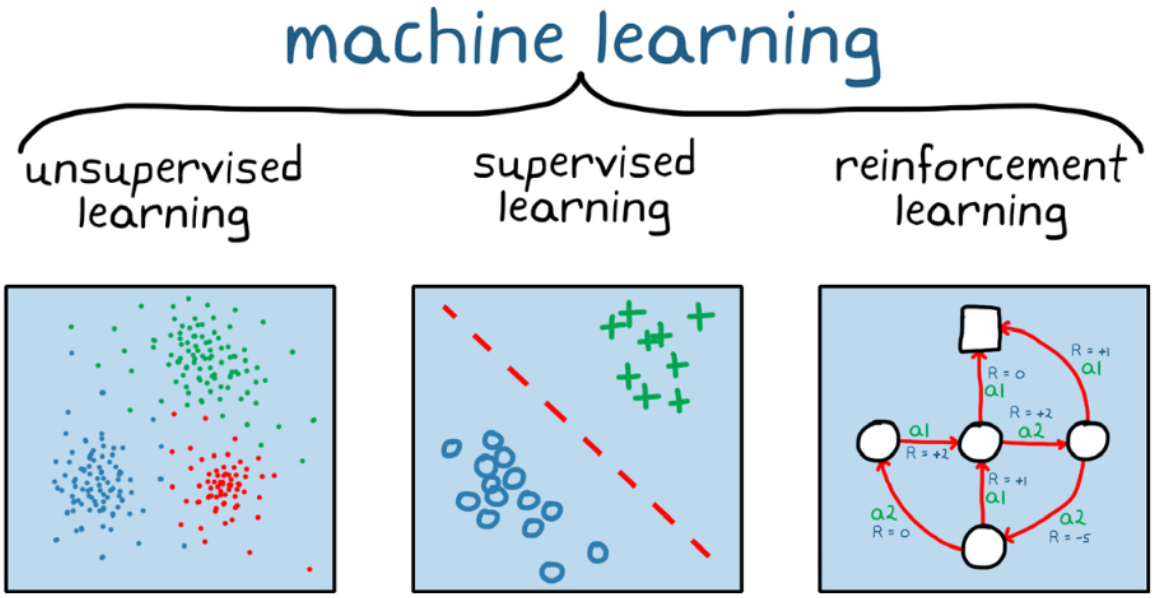
Screening og test af ny måleteknologi

State of the art – Reinforcement Learning

24. april 2023
Proj.nr. 2009615
Init.: JCAH/MT/JEPA

Indenfor kunstig intelligens (AI) findes der tre grundlæggende paradigmer:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning (RL)



Figur 1

Den mest udbredte af disse er Supervised Learning, hvor man udvikler sine modeller vha. annoteret data. Alt afhængig af opgavens sværhedsgrad kan dette resultere i en stor tidsmæssig arbejdsbyrde i selve annoteringen af data, hvis man vel at mærke gerne vil opnå en høj præcision i sine modeller. Der findes dog efterhånden mange måder at optimere denne proces på, bl.a. ved at udnytte prætrænede netværk, hvilket er netværk trænet på store datamængder fra eksempelvis ImageNet[1] eller COCO-datasættet[2], som gør dem i stand til at genkende farver, former og karaktertræk. I slagteribranchen bruges Supervised Learning bl.a. til kategorisering af produkter, bestemmelse af skærelinjer og registrering af fejl.

[1] https://www.image-net.org/

[2] https://cocodataset.org/#home

I modsætning til Supervised Learning bruges der ikke annoteret data indenfor Unsupervised Learning. Her lærer modellerne mønstre i data uden annoteret træningsdata, altså uden at vide, hvad der er rigtigt og forkert. Dermed fjernes den store tidsmæssige arbejdsbyrde i annoteringen af data. Tanken bag dette er at give modellerne de bedst mulige forudsætninger for at lære datadomænet at kende af sig selv. Der er dog mange udfordringer ved dette, og det er vanskeligt at kontrollere, hvad der egentlig foregår under 'kølerhjelmen'. State-of-the-art heri er bl.a. anomali-detektion i non-uniforme datasæt. Dette kunne fx være detektion af fremmedlegemer i produkter, hvor forekomsten af disse kun svarer til en promille af det samlede datasæt. Her ville det nemlig være svært at finde og annotere billeder nok til at kunne træne en Supervised model.

Supervised og Unsupervised Learning er velkendte og benyttes flere steder i industrien i dag. De har deres styrker og svagheder, men der findes domæner, hvor hverken det ene eller andet område er godt nok til at beskrive kompleksiteten af et problem. Og det er her, Reinforcement Learning (RL) kommer ind i billedet.

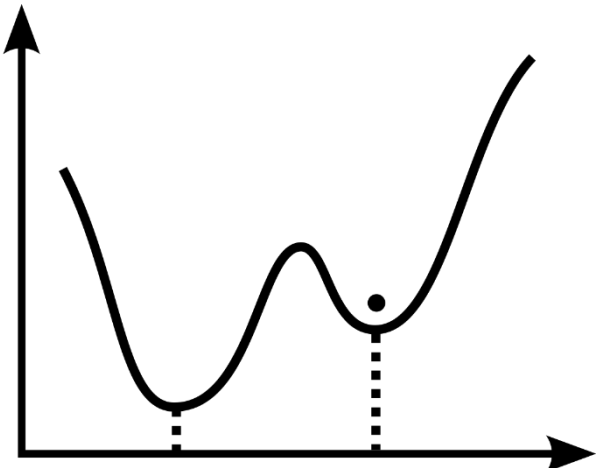
RL har vundet popularitet de seneste år pga. dets evne til at styre meget komplekse adfærdsmønstre. Et eksempel på dette ses i spilverdenen, hvor modellen AlphaZero lærte at spille brætspil, heriblandt skak, og endte med at slå flere grandmasters i skak. Derudover bruges RL også af Tesla i deres OpenAI-projekt til selvstyring af biler. Deres formål har også været at lave et framework (Gym-projektet)[3], hvor man kan benchmarke RL-modeller, da det, grundet kompleksiteten, har været svært at gøre tidligere. Med sådant et framework er RL derfor blevet mere tilgængeligt i almen dataanalyse, hvor man lettere kan benchmarke og reproducere sine resultater.

RL er et fantastisk redskab, når man har en situation eller problemstilling, hvor datadomænet kan afgrænses af nogle prædefinerede regelsæt; fx i computerspil eller skak. Der er randbetingelser, som dækker over brættet, og regler for, hvordan de forskellige brikker må bevæge sig. Disse er forholdsvis lette at definere i et program. Derimod kan det være meget svært at optimere grundet kompleksiteten i og størrelsen af udfaldsrummet. Det er netop her, RL kommer ind i billedet, da den optimerer sig selv. Samtidigt kan den ovenikøbet spille mod sig selv og lære af det. Man kan derfor også forestille sig situationer i den virkelige verden, hvor man kan lave disse prædefinerede "spilleregler", såsom ved automatisk styring af en robot, hvor den gives et specifikt formål at optimere imod. Dertil er der selvfølgelig nogle andre udfordringer, såsom at skaffe sig den nødvendige mængde data at optimere sin model efter. Til gengæld giver det også nogle friheder, da man ikke er begrænset af at bruge data fra den virkelige verden. Da man i princippet skal lære et miljø at kende, kan dette sagtens gøres i en simuleret verden. Dette giver nogle fordele som fx uanede mængder data, og man kan lettere indføre situationer, der ikke sker så ofte i den virkelige verden. Der vil dog altid være scenarier fra den virkelige verden, som er svære at simulere. Og i sidste ende skal ens netværk trods alt køre i virkeligheden. Hvordan man vælger at træne sin model, afhænger derfor også helt af, hvilken problemstilling man prøver at optimere. Fx vil det være smart at starte træningen af en selvstyret robot i et simuleret miljø, da man undgår, at robotten går i stykker, hver gang den falder. Derefter kan man fortsætte træningen i et virkeligt miljø, når robotten har lært at holde sig oprejst og støtte sig selv i et simuleret miljø.

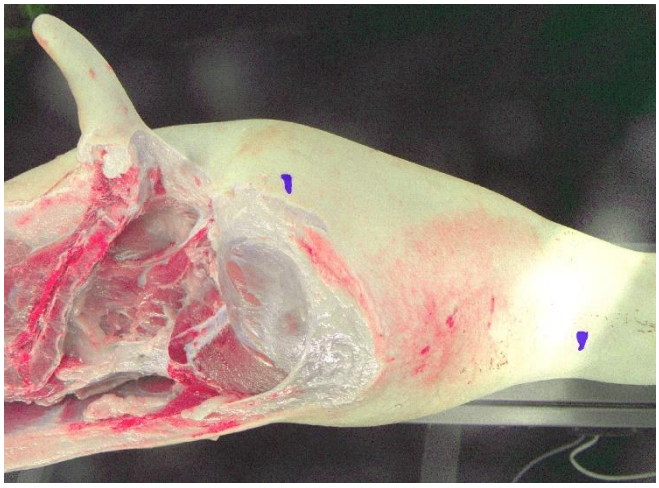
Nogle af de vigtigste betegnelser inden for RL er følgende:
1. En agent, der skal lære.
2. Et miljø/problem, som agenten skal lære i/lære at løse.
3. Et action space, som udgør de beslutningsmuligheder/træk agenten har.
4. En policy, som agenten følger for at træffe beslutninger.
5. En reward, som agenten observerer efter hver beslutning.

[3] https://github.com/openai/gym

Hvis man forestiller sig problemstillingen, at en robot skal rengøre en slagtekrop, så vil robotten være agenten. Miljøet er selve slagtekroppen og dets omgivelser. Action space vil være mulighederne for robotten for at bevæge sig op, ned, frem, tilbage og til siden. En reward kunne være at belønne robotten for at fjerne en gødningsklat og straffe den for at overse en gødningsklat. Policy er selve strategien for, hvordan man lærer. Det kunne være at gentage sig selv, hver gang man fik en belønning, eller at hver fjerde beslutning vælges tilfældigt for at afsøge hele udfaldsrummet og dermed opnå fuld forståelse for miljøet. De fleste policy indeholder elementer af tilfældighed, da man derved i større grad udforsker hele miljøet. Man kan illustrere det som vist i figur 2. Her ligger kuglen i det falske minimum, og hvis ens algoritme ikke er justeret til at udforske forskellige muligheder, og altid starter med at rulle ned fra højre side, vil den potentielt aldrig ramme det sande minimum, som ligger til venstre. Dette ville ikke være tilfældet, hvis man tilfældigt lod kuglen falde ned fra enten højre eller venstre side, eller nogle gange midtfor. Selve implementeringen af forskellige policy er gjort let tilgængelig i Stable-Baselines3-projektet[4].



Figur 2

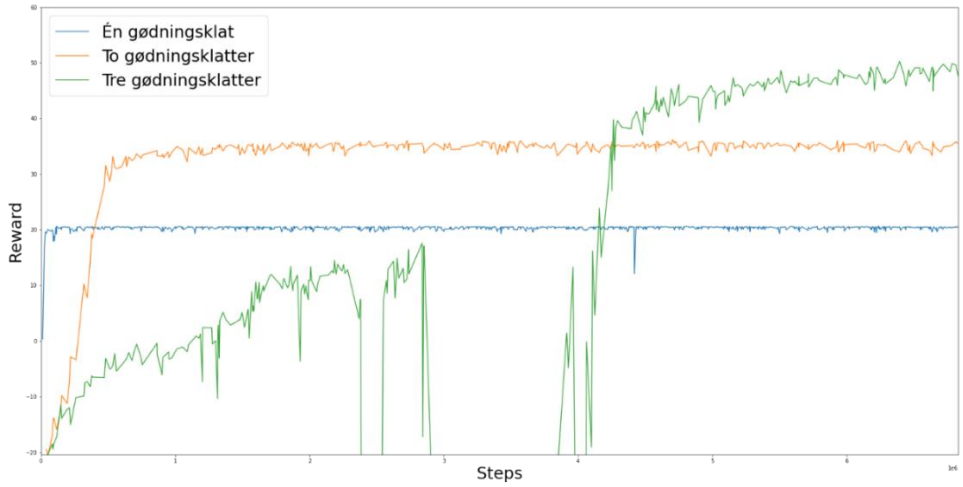


Figur 3

For at undersøge, hvor let det er at træne en RL-model, har vi fokuseret på scenariet med en robot, der skal rengøre en slagtekrop (eller en del af den). For nemheds skyld har vi set det som et problem i to dimensioner. Vi har malet et begrænset antal gødningsklatter på forskellige billeder af grise og antaget, at vi har en robot med en dampsuger af en vis størrelse. I figur 3 ses et eksempel, hvor gødningsklatterne er malet blå (for illustrationens skyld).

Vi har trænet modeller til at rengøre kroppe med hhv. en, to og tre gødningsklatter. Vi har givet robotten en tilfældig valgt startposition hver gang og givet den muligheden for at bevæge sig op, ned, til højre og til venstre. Da vi ønsker, at robotten fjerner al gødning på kortest mulig tid, giver vi en lille straf til robotten, hver gang den bevæger sig. Hvis den bevæger sig tættere på en gødningsklat, gives den dog en mindre straf, end når den bevæger sig længere væk fra en gødningsklat. Derudover giver vi robotten en stor straf, hvis den bevæger sig ud af billedet, og en stor belønning, hver gang den fjerner en gødningsklat. Illustrationen i figur 4 viser, hvor mange iterationer der skal til, før modellerne begynder at konvergere til en optimal løsning.

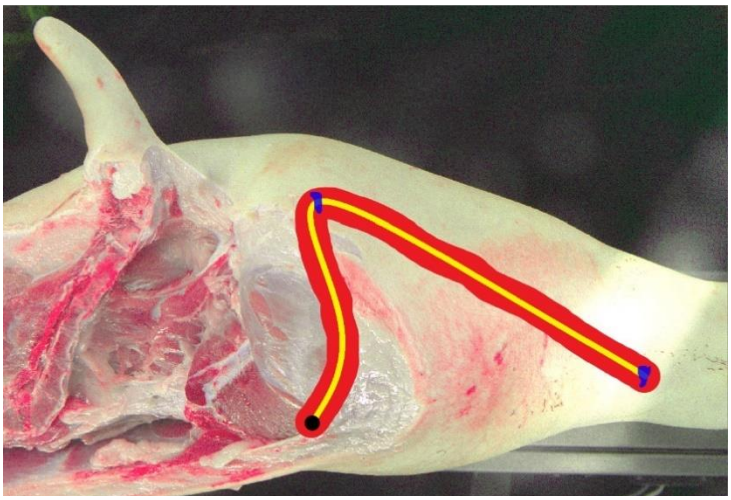
[4] https://jmlr.org/papers/volume22/20-1364/20-1364.pdf



Figur 4

Det ses tydeligt, at ved én gødningsklat konvergerer løsningen hurtigt efter meget få steps og modtager næsten altid den maksimale reward. Der går noget længere tid for modellen, der skal fjerne to klatter, og endnu længere for modellen, der skal fjerne tre gødningsklatter. I praksis vil en model selvfølgelig skulle trænes til at håndtere et arbitrært antal gødningsklatter, hvilket tager endnu længere tid, da udfaldsrummet herved forstørres markant. Alle modeller er trænet med OpenAI's PPO-policy[5]. Et eksempel på modellens 'rute' til rengøring af en gris med to klatter er vist i figur 5.

Den sorte prik er startpunktet, og den gule linje er robottens rute, hvor det røde angiver størrelsen af dampsugeren, dvs. det område som reelt set rengøres. Man kan se, at den ikke går den helt direkte vej hen til første gødningsklat, så løsningen kunne være bedre, men dette var også mere ment som en lille illustration og for at lære brugen af RL. Derudover kunne man sikkert også have lavet en bedre løsning – endda meget hurtigere – ved at bruge mere traditionelle metoder.



Figur 5

Og sådan vil det nok være for de fleste problemer, som skal løses i industrien. Hvis man kun er interesseret i ydeevne, vil det i langt de fleste tilfælde være bedre at anvende mere traditionelle metoder. Samtidig har man lidt bedre styr på, hvad der egentlig foregår i modellen. I fremtiden vil der dog nok komme flere scenarier, hvor fx robotter skal tage beslutninger i mere komplekse miljøer, hvor udfaldsrummet er markant større end i dag. Og her kunne man godt forestille sig, at det vil være tiden værd at undersøge, hvordan RL kunne hjælpe med de problemer. I løbet af de kommende år vil udviklingen og tilgængeligheden af RL-programmer nok også tage nogle store spring, der kan gøre det endnu hurtigere og lettere at træne modeller. Det vil derfor være værd at holde øje med, hvad der sker indenfor udviklingen af Reinforcement Learning i den kommende tid.

[5] https://openai.com/blog/openai-baselines-ppo/