eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
VITORIA-GASTEIZKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE VITORIA-GASTEIZ

INDUSTRIA ELEKTRONIKOAREN

ETA

AUTOMATIKAREN INGENIERITZAKO GRADUA

GRADU AMAIERAKO LANA

IKASKUNTZA SAKONA (*DEEP LEARNING*) TEKNIKEN ANALASI TEORIKOA ETA PRAKTIKOA MATLAB-EN.

Egilea:

Erik Gorospe Hernaez

Zuzendaria:

Ekaitz Zulueta Guerrero

VITORIA GASTEIZKO INGENIERITZA ESKOLA

2023

ESKERRAK

Eskerrak, Ekaitz Zuluetari proiektu guztiaren inguruan egindako lanagatik. Proiektuan zehar pazientziarekin guidatu nau, une oro nire zalantza eta galderei erantzunak bilatuz eta *"Reinforcement Learning"* mundua nire begitara zabaldu du.

Mila esker, Teo Ricori unibertsitatea aukeratzeko orduan aholkuak emateagatik, oso pozik nago egindako hautaketarekin. Horretaz gain, eskerrak, urtero izandako zalantza askotan laguntza eskaintzeagatik eta gradu amaierako lanaean hasierako guidapen bat emateagatik.

Eskerrik asko, Euskak Herriko Unibertsitateari Matlab lizentzia emateagatik; Izan ere, lanean erabilitako herramientarik garrantzitsuena izan da.

Análisis teórico y práctico a través de MatLab del aprendizaje profundo *(deep learning)*.

Resumen

El aprendizaje por refuerzo es una técnica de prueba y error basada en un sistema de recompensas. El agente, a cargo de cumplir el objetivo propuesto, interacciona con el entorno. En cada interacción el agente recibe una recompensa que representa cómo de buena ha sido la acción realizada. De esta manera, después de un proceso de aprendizaje, el agente es capaz de encontrar por sí solo la solución óptima del problema.

Dentro de esta técnica se han desarrollado muchos algoritmos, los cuales, sirven para diferentes situaciones. Hay que distinguir entre políticas, agentes y críticos deterministas o estocásticos. De esta manera, se puede elegir uno de tantos algoritmos como: PPO, DDPG, SAC, TD3…

Gracias a programas como Matlab somos capaces de trabajar con estos algoritmos de una manera sencilla y ponerlos a prueba en diferentes ámbitos como la conducción automática o la distribución de probabilidad.

Ikaskuntza sakona *(deep learning)* tekniken analisi teorikoa eta praktikoa MatLab-en.

Laburpena

Indartze ikaskuntza froga eta errore teknika bat da, sari-sistema batean oinarritua. Agenteak, arazoari irtenbide optimo bat lortzeko helburuarekin, ingurunearekin elkarreragiten du. Interakzio bakoitzean, agenteak sari bat jasotzen du, egindako ekintza zein ona izan den adierazten duena. Horrela, ikaskuntza-prozesu baten ondoren, agentea gai da arazoaren konponbide optimoa bere kabuz aurkitzeko.

Teknika horren barruan algoritmo asko garatu dira, eta egoera desberdinetarako balio dute. Politikak, eragileak eta kritiko deterministak edo estokastikoak bereizi behar ditugu. Horrela, algoritmo hauetako bat aukera daiteke: PPO, DDPG, SAC, TD3…

MatLab bezalako programei esker, algoritmo horiekin modu errazean lan egiteko gai gara, eta hainbat esparrutan frogatzeko ahalmena dugu, hala nola gidatze automatikoan edo probabilitate-banaketan.

Aurkibidea

1. SARRERA ETA HELBURUAK

Indartze ikaskuntzaren oinarriak eta teoriaren garpaena Richard S. Sutton eta Andrew G. Barto ikerlariek planteatu zuten *"Reinforcement Learning: An introduction"* liburuan [1]. Ikertzaileek kontzeptu honen oinarrian neurozientzia eta animal portaera daudela ezartzen dute, hauei formalizazio matematiko bat emanez. Horretaz gain, *Reinforcement Learning*-ek ingenieritzaren inguru askorekin erlazioa daukala onartzen dute, halere, garrantzitsuenak kontrol-ingenieritza, kontrol optimoaren teoria eta programazio dinamikoa matematikan edo informatikan dira.

1.1. ZERTAN DATZA INDARTZE IKASKUNTZA?

Reinforcement Learning esperientzian, inguruarekin elkarrekintzan eta saiakuntza anitzen akatsetan (prueba y error) oinarritzen den ikaskuntza da. Ideia hau adimen artifizialetan inplementatzea, biologiatik dator; Izan ere, munduan animali guztiek bizitzan zehar modu honeta ikasten dute.

Teknika honekin gain hartze diren dilema guztietan helburua bera da: aukera optimoak aurkitzea, dilemari irtenbide onena aurkitzeko.

Denok dugu jokabide inkontzienteak ikasi izanaren esperientzia. Adibidez, auto bat gidatzen ikasten ari garenean, gure arreta guk egindako ekintzak ingurunean duten eraginari doa, ondoren, eragin hori baloratzeko. Hau da RL-en oinarria. Ikasketek [2], erabiliko ditugun ikaskuntza algoritmoak gizakiak ikasteko erabilitako mekanismoekin erlazionatzen dituzte.

Ikasten duen programari ez zaio erabaki zuzenak noiz eta non hartzeko programatzen. Honen ordez, sarietan (*reward*-etan) oinarritzen den sistema bat eraikitzen da[1], non, egoera partikular batean egin daitekeen ekintza onena sari handiena izango du. Agentea saiakerak eginez eta askotan huts egin ondoren, pixkanaka pixkanaka erabaki onak hartzera bideratuko da, hau da, sari handienak dituen bidea hartuko du, amaieran, erantzun hoberenak momentu oro aukeratuz.

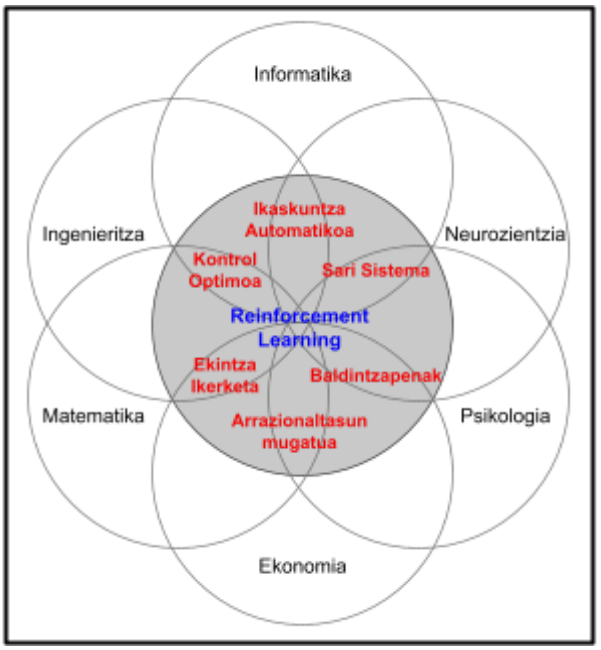
Esaterako, bi artikulazio eta amaiera matzarda bat duen beso robotikoa egin ditzazkeen mugimenduak kalkulatu daitezke alderantziko zinematika (cinematica

[1] Paulov-ek erreflexu ikasiaren edo erantzunaren ikasketekin antzekotasuna du [3].

inversa) erabiliz, kalkulu hauen zailtasuna robotaren konplexutasunarekin batera exponentzialki handitzen dira. Kasu hoberenean, kalkulu guztiak egin ondoren, robotaren posizioa (estatika), mugimendua (zinematika) eta indarra (dinamika) kontrolatu genezake, halere, metodo zaharkitu honekin denbora asko behar izango da eta portaera inteligentea simulatzea oso zaila bilakatuko da.

RL-rekin gure eskura, arazoa sinpleagoa bilakatzen da. Soilik, beso robotikoa bere baitan mugitu daitekeen inguru bat diseinatuz eta sarietan oinarritutako sistema bat eraikiz, robota bere baitan ikas dezake portaera egoki bat izaten. Halere, sarietan oinarritutako sistema eraikitzea ez da hain erraza ez badakizu lortu nahi dituzun helburuak eta neurtu ahal diren aldagaiak zehazki zeitzuk diren. Hau da edukiko dugun ataz garrantzitsuena eta zailena.

Zientzia askoren nahasketa da indartze ikaskuntzaren kontzeptua **1.irudian** egindako Venn diagraman ikus daitekeen moduan.



1.Irudia: Venn diagrama, RL (*Reinforcement Learning*) bateratzen dituen zientzia guztiekin [4].

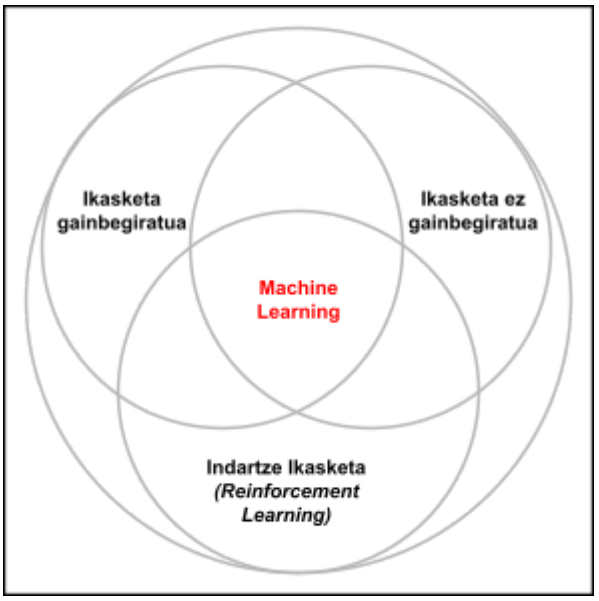
1.1.1. Adimen artifizialarekin eta beste adarrekin duen erlazioa.

Ikasketa automatikoa edo *Machine Learning* erantzuna bilatzeko algoritmo elkareragileak (interaktiboak) erabili ahal diren arazoetan aplikatzen da. Algoritmo hauek bide eskutuak, datu patroi eskutuak edo tendentziak aurkitzen eta ikasten dituzte. Ezaugarri guzti hauek, arazoari aurre egiteko eta erantzuna ateratzeko modelu sinple bat eraikitzea ahalbideratzen digute. Bi modu nagusitan eman daiteke ikasketa hau: **ikasketa gainbegiratua** edo **ez gainbegiratua**.

Ikasketa gainbergiratuan bi datu mota daude. Alde batetik, entrenamendu datuak guztiz klasifikatuta eta erantzuna aterata daude. Hauekin gure makina ikasketa hasten du, ateratako erantzunak guk jarritakoekin konparatuz. Bestetik, egiaztatze datuak daude. Hauek ikasketa guztiz bukatu ondoren erabiltzen dira eta ikasitako politika (aurrerago azalduko dugun kontzeptua) egokia den frogatzeko erabiltzen dira.

Gainbegiratzen ez den ikasketan, aldiz, dautak ez dira bitan banatzen eta algoritmoa da patroiak eta tendentziak aurkitzen dituena, aurrerago, politika zentzudun eta optimo bat eraikitzeko.

Indartze ikasketa (aprendizaje por refuerzo) gainbegiratzen ez den ikasketaren itxura du, datuak ez direlako bitan sailkatzen. Dena den, RL-rekin konpontzen diren arazoetan, egoera eta ekintza sekuentzia garrantzi handia hartzen du.



2.Irudia: Venn diagrama, RL (*Reinforcement Learning*) eta ML (*Machine Learning*) duten erlazioa irudikatuz [4].

2.Irudian ikus daitekeen moduan RL-k azaldutako lehenengo bi kontzeptuekin ezaugarriak partekatzen ditu, baina kontzeptu berri bat da. Indartze ikaskuntzaren barruan sare neuronalen erabilera zabaldu da azken hamarkadan, kontzeptu berri bat guztien begietan jarriz "*Deep Machine Learning"* [5].

1.1.2. Agentea eta ingurua.

RL teknika aplikatuz irtenbidea ateratzen zaion arazo batean, gure robota errepresentatzen duen kodigo/programazioari **Agentea** deitzen zaio. Agentea berarekin eta inguruarekin interakzionatzen du ekintzen bitartez.

Ingurua egin beharreko atazaren errepresentazioa da. Inguruak agenteak aukeratutako **ekintzei erreakzionatzen** du, honela agentearekin komunikatuz. Ekintza hauen ondorioz, ingurua aldatzen da bere **egoera** modifikatuz. Agenteak egoera hau ikus eta baloratu dezake.

Ingurua aldagaiez osatuta dago. Aldagai bakoitza agentea entrenatu behar dugun atazan garrantzi bat izan beharko du. Esaterako, esan dezagun lehen aipatu dugun beso robotikoa objetu bat mahai baten alde batetik bestera mugitu behar duela. Seguraski, ingurua osatuko duten aldagaiak mahaiaren eta objetuaren tamaina, forma eta abar izango dira, baita egon daitezkeen ostopoak ere. Baina, mahaiaren salneurria edo obejtuaren kolorea ez dute atazan garrantzirik izango, horregatik, ez ditugu inguruaren barnean sartuko.

Aldagai garrantzitsu guzti hauek eta hartu ditzazketen balio guztiak, **espazio bektorial**[2] bat osatuko dute, honi **egoera-espazioa** deituko diogu. Egoera bakoitza, aldagaien konbinazio bakarra errepresentatzen du inguruaren momentu espezifiko batean (t).

Askotan, agenteak ez du guztiz inguruaren egoera ikusten, hautematzen dituen aldagaiak **behaketak** (Observaciones) deituko ditugu. Hauek beste espazio bektorial bat osatzen dute, **behaketa-espazioa** deitzen dena. Batzutan behaketa-espazioa eta egoera-espazioa berdinak direla aipatzea merezi du.

---

[2] bektore espazioa hutsa ez den multzo batetik sorturiko egitura aljebraiko bat da, egitura hau aipatutako multzo ez hutsa horren eta bektore batuketa batetik (barne operazioa) edota eskalar biderketa sortzen da [6].

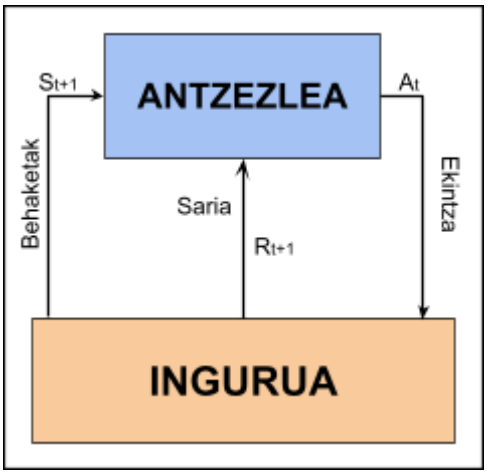
Sarietan oinarritutako sistema batean gaudela aipatu dugu, baina zer da **sari (*reward*)** hau? Zer errepresentatzen du? Sariak agenteak hartutako ekintza genuen egoeran zein ona izan den, hau da, erantzun optimora zenbat hurbildu garen, estimatzen duen balio numeriko eskalarra da. Beste batzutan, zigorretaz hitz egin dezakegu. Hau lehen deskribatutakoaren berdina da, baina balio negatiboak (penalizazioak) erabiliz. Saria funtzio baten bidez kalkulatzen da, honi **sari-funtzioa (r*eward function*)** deitzen zaio. Laburbilduz, inguruaren egoera eta hartutako ekintza zenbaezten duen sari-funtzio egoki bat ezartzen badugu, amaieran, ekintza hoberena aukeratuko dugu egoera guztietan.

Sari guztien batuketari **Itzulera (retorno)** deitzen zaio. Kontuz, hau sari indibidualekin (*reward-arekin*) ez nahastea, bibliografiaren artikulu askotan gertatzen den moduan. Itzulera erabiltzen da soilik amaierako egoera garrantzia duen arazoetan, lehiaketa moduko arazoetan, non, askotan garrantzia duen gauza bakarra irabaztea da. Irabazteak itzulera postibo bat izango du eta galtzeak, aldiz, negatiboa.

1.1.3. Ikasketa metodoa

Ikus dezagun **3.irudia**, non, RL-k jarraitzen duen ikasketaren simplifikazioa bat ematen da. Ikasketa buklean gertatzen da, agentea ataza bukatutzat eman arte edo guztiz huts egin arte.

Ikasketan simulazio asko (bukle asko) egiten dira eta saiakera bakoitzari **kapitulua (episodio)** deitzen zaio, kapitulu bat bukatutzat ematen da agentea ataza bukatu duenean edo guztiz huts egin duenean. Huts egindako kasuetan sariaren balioa ez da ona izango. Kontuan izan behar dugu, agentea ez duela soilik arrakastetatik ikasten eta akatsak ikasketa prozezuaren atala izango direla.



3.Irudia: RL ikasketa prosezuaren eskema sinplifikatua.

Kapitulu bakoitza pausu asko iraugo ditu, non, une oro ingurua eta agentea elkar komunikatzen egongo dira. Pausu bakoitzari ***"step"*** edo *"t"* deituko diogu eta agentearen eta inguruaren arteko interakzio bat adieraziko du.

"Step" bakoitzean, hurrengo sekuentzia buruzten da:

- Agenteak behaketak ikusten (S_t) ditu eta inguruaren egoera (E_t) aztertzen du.
- Agenteak ekintza bat (A_t) egiten du, jasotako informazioaren baitan.
- Ekintza inguruaren egoera aldarazten du (E_{t+1}) eta saria (*reward-a*) (R_{t+1}) kalkulatzen da.
- Agenteak egoera berria eta sari berria ikusita hurrengo ekintza (A_{t+1}) aukeratzen du, buklea berriro hasiz.

1.1.4. Politika

Politika edo plana agentearen portaera definitzen du inguruaren egoeren aurrean. Agentearen joera aurre ezarritako plan batean oinarritzen da. Politika funtzio matematiko batean errepresenta daiteke, inguruaren egoera jakin batean agenteak aukeratuko duen hurrengo ekintza definitzen duena.

Entrenamendu gabeko agente baten politika uniformea da, hau da, egoera partikular batean ekintza guztien probabilitatea bera da. Dena den, entrenamenduak agentearen politika funtzioa aldatzen/optimizatzen du ekintzen probabilitatea aldatuz, hau

hobeagotzeko. Hasieran, agenteak egindako ekintzak ausazkoa izango dira eta jasotako sarien bitartez determinatuko du ekintzen balioak egoera anitzetan. Kapitulu batzuen ondoren, agenteak ikasiko du ekintza egokia hartzen egoera bakoitzean, hots, bere politika optimoa izango da.

Bi taldetan sailkatu daiteke agente baten politika: **deterministikoa** edo **estokastikoa**. Inguru simple eta diskretuetan politika taula batean irudikatu daiteke, non, estatu bakoitzean hartu beharreko aukera ager daiteke. Taula osatu daitekenean **politika deterministikoa** izango da.

Alabaina, normalean **politika estokastikoa** erabiltzen da. Honetan, agente batek egin ditzazkeen ekintza multzoa ez da finitua, infinitua baizik. Ondorioz, probabilitateaz lagunduz agenteak inguruaren estatu konkretu baten aurrean, egin beharreko ekintza aukeratzen du [5]. Agenteak entrenamenduan egoera berean ez du une oro ekintza bera egingo eta probabilitate handiena duten ekintzen artean bat aukeratuko du, amaieran, bakarra ezarriz egoki moduan.

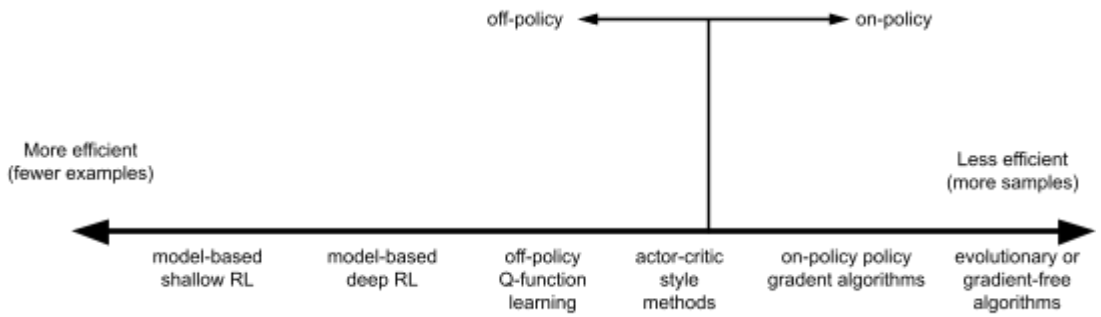
Politika estokastikoen barnean gure ekintza-espazioa diskretua bada, agenteak **politika kategorikoa** duela esango dugu, hala eta guztiz ere, ekintza-espazioa jarraitua bada gure **politika gaussiarra** izango da, batez besteko (μa) eta desbideratze estandar (σ) batez definitua.

1.2. ALGORITMOAK

RL-n ez dago soilik ikasketa algoritmo bakar bat. Behekaldean ikus daitezkeen bi irudiak (**4.irudia** eta **5.irudia**) bi sailkapen egiten dituzte.

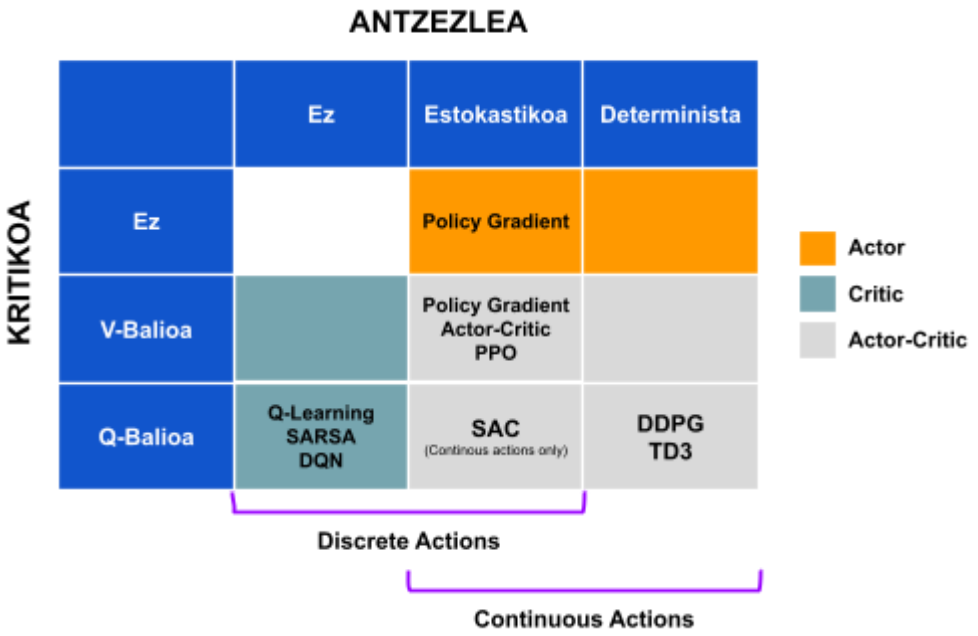
Alde batetik, galdera bat proposatu dezakegu. Zenbat kapitulu/lagina behar dira politika egoki bat eraikitzeko? **4.Irudian** RL-k dituen algoritmoak efizientziarekiko eta lagin kantitatearekiko klasifikatuta daude. *"Off-policy"* algoritmoek, lagin berri bat sortu gabe politika aldatu dezakete, *"on-policy"* algoritmoek, aldiz, lagin berri bat sortzen dute politika aldatzen duten bakoitzean.

Efizientziak konputazio denbora txikiagoa ekartzen du, baina konputazio ahalmen handiagoa.



4.Irudia: RL algoritmoen sailkapena efizientziarekiko eta lagin kantiteareiko.

Beste alde batetik, antzezle eta kritiko (aurrerago azalduko dugun bi kontzeptu) desberdinen artean, bigarren sailkapen bat daukagu; Izan ere, lehen azaldu dugun moduan haien politika bitan sailka daiteke: **determinstikoa** eta **estokastikoa**.



5.Irudia: RL- algoritmoen bigarren sailkapena antzezle eta kritikoarekiko.

1.3. HELBURUA ETA METODOLOGIA

Lan honetan teorikoki *Deep Reinforcement Learning-en* teknikak aztertuko eta alderatuko dira, azalpen teoriko eta praktiko logiko bat euskaraz emateko. Algoritmo hoietariko batzuk *Deep-Q-Network*, *Soft Actor-Critic* edo *Proximal Policy Optimization* izan daitezke.

Azterketa teorikoaren oinarriak, *"Reinforcement learning: An introduction"* liburutik hartu ditugu [1]. Honek erabiliko ditugun algoritmo guztien zimeduak ezarri zituen eta egilearentzako hasieran gai hau berria zenez, hasteko egokia zela ezarri genuen.

Hurrengo pausuak gauzatuko dira, azterketa teoriko eta praktiko honetan:

1. *Reinforcement Learning*-en azalpen teorikoa egingo dugu.
2. Balio/Sari funtzioen erabilera azalduko dugu.
3. Sare Neuronalak aztertu eta adibideak ezarriko ditugu.
4. Indartze ikaskuntzaren barnean dauden hiru familia handiak azalduko ditugu.
    a. *Value-based* metodoa
    b. *Policy-search* metodoa.
    c. *Actor-Critic* metodoa.
5. *Actor-Critic* metodoaren barruan algoritmo desberdinak banaduko eta azalduko ditugu.
    a. *Deterministic Policy Gradients*.
        i. Adibidea bat eginez Matlab-en.
    b. *Proximal Policy Optimization*.
        i. Adibidea bat eginez Matlab-en.
    c. *Soft Actor Critic.*
    d. *Twin Delayed Deep Deterministic.*
6. Bukatzeko, ikasketa guztiaren ondoren, ateratako ondorioz azalduko ditugu.

2. *REINFORCEMENT LEARNING*

RL teknikarekin irtenbidea aurkitzen zaien arazo guztiak Markov-en erabaki hartze prosezu baten moduan errepresenta daitezke [1].

2.1. MARKOV-EN ERABAKI HARTZE PROSEZUA

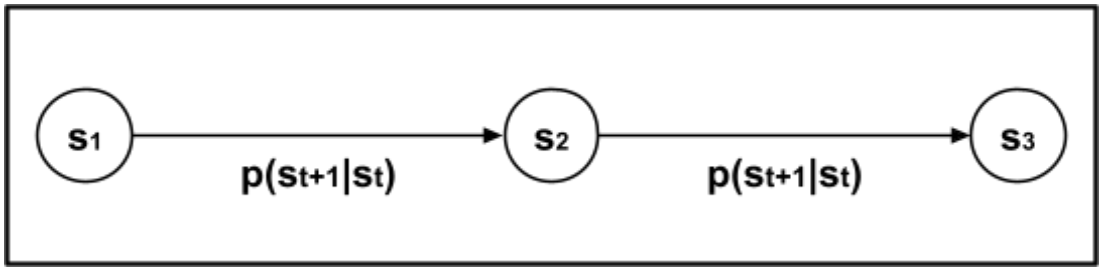
Prosezu honen ideia: etorkizuna iraganaren aske egotea da, iragana orainaldian isladatuta dagoelako. Etorkizuna soilik orainaldia baldintzatuko du.

Matematikoki etorkizun baten probabilitatea honela adierazten da:

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1,\ldots,S_t] \quad (1)$$

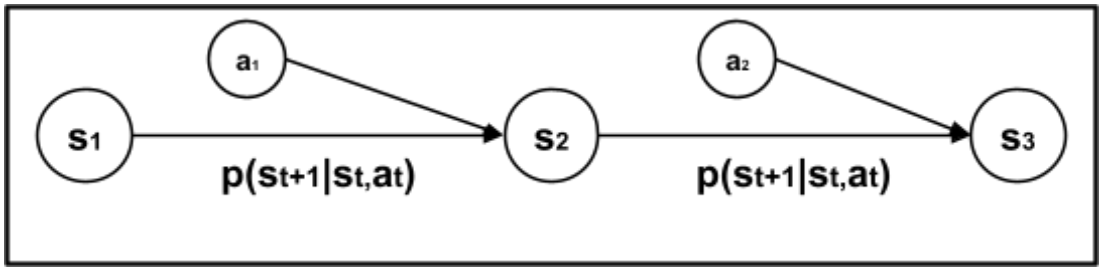
1.Ekuazioan ikus daitekeen moduan, egoera bat berarekin darama beharrezkoa den informazio guztia hurrengo egoerara igarotzeko.

Ondorioz, logika hau jarraituz egoera sekuentzia bat irudika daiteke **6.irudian** bezala.



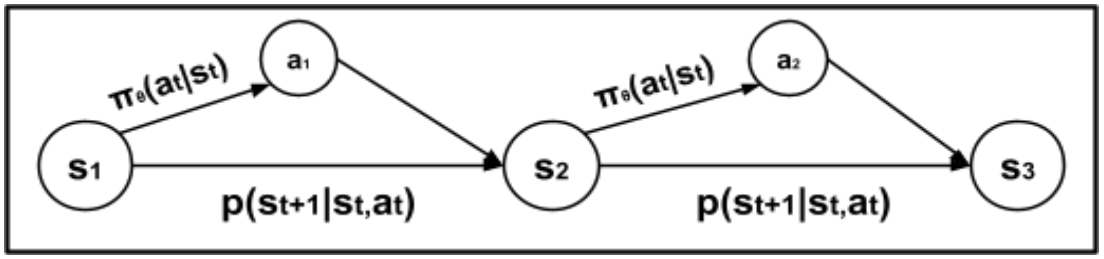
6.Irudia: Markov-en sekuentzia batean trantsizio baten probabilitatea soilik etorkizuna eta orainaldia baldintzatzen dutelaren adierazpena.

Honi erabakiak (a_t) gehitzen badira, erabaki sekuentzia bat sortzen da. Ikus **7.irudia.**



7.Irudia: Markov-en erabaki hartze prosezuaren adierazpena ekintzekin.

Bukatzeko, gogora dezagun RL teknika erabiliz politika optimo bat bilatzen dela. Hasieran aipatu dugun moduan, politikak egin daitezkeen ekintza guztien artean probabilitate banaketa bat egiten du. Horregatik, Markov-en erabaki sekuentzia, RL teknika erabiliz, **8.irudia** isladatuko du.



8.Irudia: RL-k duen sekuenetzia Markov-en erabaki prosezu baten modura.

2.2. BALIO FUNTZIOAK.

2.2.1. Egoera balio funtzioa edo V.

Egoera balio-funtzioa, bere izena adierazten duen moduan, inguruaren egoera baten balioa ematen du, politika (π) bat jarraitzen dugunean. Gehienetan, **egoera balio-funtzioak** ingurua duen egoera-sekuentzia baten balioa ematen du, ez soilik egoera bakar batena. Matematikoki adierazita **2.ekuazioan** ikus daiteke.

$$v_{\pi}(s) = E_{\pi}[G_t|S_t = s] \quad (2)$$

2.2.2. Ekintza balio funtzioa edo Q.

Hartutako erabakiaren balio-funtzioa edo Q-funtzioa, izena berak esaten duen moduan, erabaki baten balioa ematen du, politika batez deteminatuta, inguruaren egoera konkretu batean. Matematikoki adierazita **3.ekuazioan** ikus daiteke.

$$Q_{\pi}(s_t, a_t) = E_{\pi}(G_{\pi}|S_t = s, A_t = a) \quad (3)$$

Funtzio hau maximizatzea RL algortimo askoren helburua da.

Qπ matematikoki determinatzea oso zaila da, halere, hurbilketak egiteko moduak daude.

2.2.3. V funtzioa eta Q funtzioaren erlazioa.

V eta Q funtzioa elkarren artean erlazionatuta daude.

Ingurune estokaistiko batean, egoera baten ekintza guztien probabilitateen batura 1 da. Ondorioz, erabaki guztiak duten balioa (Q futzioarekin kalkulatu dezakeguna) bere probabilitatearekiko multiplikatuta batzen baditugu egoeraren balioa emango digu, hau da, V-funtzioren berdina (Ikus **4.ekuazioa**).

$$V_{\pi}(s) = \sum_{a} \pi_{\pi}(a|s) Q_{\pi}(s,a) \tag{4}$$

2.2.4. V funtzio eta Q funtzioaren balio praktikoak.

Nola hobetu dezakegu gure politka (π), Qπ-ren balioa badakigu?

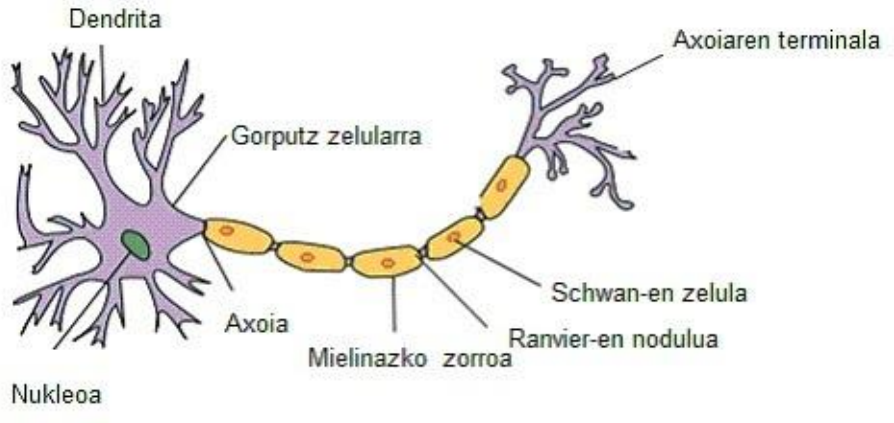
Alde batetik, Qπ kalkulatu ahal badugu, soilik aurkitu behar da balio hau maximizatzen duen politika. Qπ askotan hurbilketaz kalkulatzen denez, hau ere hurbilketa bat izango da.

Bestetik, gradienteak erabili ditzazkegu erabakien (a) balioa hobesteko. Qπ(a,s)-ren balioa Vπ(s)-ren balioa baino handiagoa bada, hartutako erabakia zeuden erabaki guztien artean hoberena da. Honela, politika modifikatu dezakegu hartutako erabakiaren balio handitzeko.

2.3. SARE NEURONALAK.

Sare neuronalen kontzeptua *Machine Learning* teknikak handitzen ari ziren momentu berean popularizatu zen, baina lehen aldiz 60ko hamarkadan entzun zen Marvin Minsky y Seymour Papert *"percepetron"* ideiari buruz egindako publikazio batean [7]. Neuronen portaera imitatzen duen elementu teorikoa da **"*percepetron*"**.

Neurona bat estimulatzean, dituen dentritetik egiten da. Horretan tentsio igoera bat ematen da 35mV-ak gainditzen duena. Honek tentsio hori jasota, 69mv eta 80mv artean dagoen tentsio pultso bat eragiten du axon-aren zehar (ikus **9.irudia**). Axon-a beste neurona batzuen dentritekin konektatuta dago.



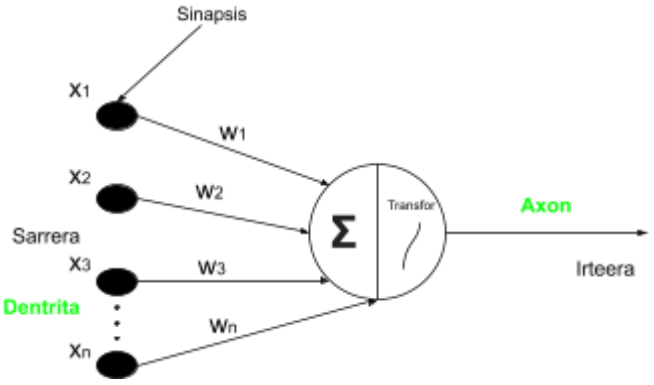
9.Irudia: Neurona baten atalak.

"*Percepetron*" elementua jokaera hau imitatzen du **aktibazio funtzio** batekin. Funtzio bat baino gehiago daude jokaera hau errepresentatzeko, "sigmoide" funtzioa (**5.ekuazioa**) da garrantzitsuenetariko bat:

$$sigm(x) = \frac{1}{1+e^{-x}} \tag{5}$$

Funtzio hau, balio negatibo guztiak 0-ra hurbiltzen ditu eta balio positibo guztiak 1-era. Horretaz gain, deribatzeko erraza da: $f'(x) = f(x)(1 - f(x))$.

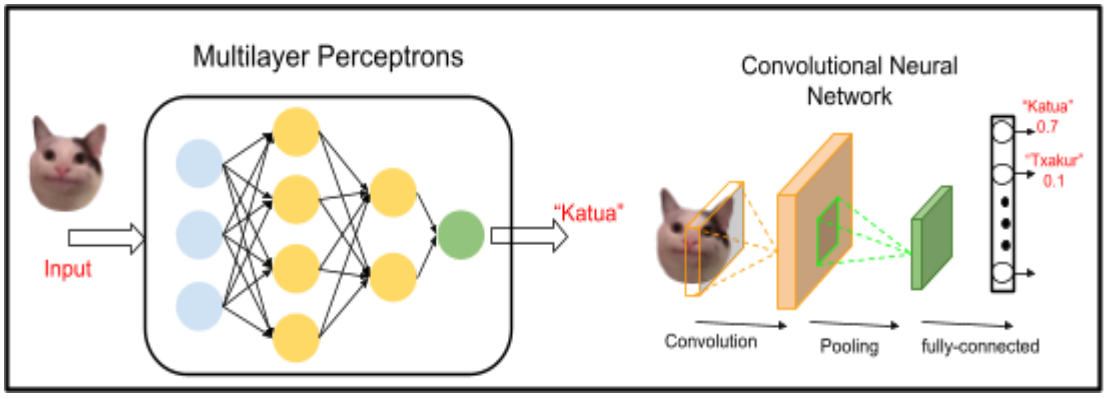
Perceptron-ak neurona-birtualak deitu daitezke. Neurona birtual hauek geruzetan multzokatzen dira, non, normalean geruza baten neurona guztiak hurrengoaren neurona guztiekin konektatzen dira. Konexio mota honi **"MLP"** ***"Multy Later Perceptron"*** (Ikus **11.irudia**) deitzen zaio. Konexio bakoitza zenbaki bat dauka, **pisua** deitzen dena. Ondorioz, "*perceptron*" baten aktibazio funtzioa, aurreko "*perceptron*" guztien balioak bider sartzen den konexioaren pisua eta guzti hauen gehiketa da (ikus **10.irudia** eta **6.ekuazioa**).



10.Irudia: "Perceptron" baten eredua.

$$AF = \sum_{t=1}^{n} (x_t * w_t) \tag{6}$$

Machine Learning-en gure agentea ikasi dezan, sare neuronal hauen pisuak aldatu behar dira, hau egiteko erabiltzen den teknika *"backpropagation"* deitzen da. Teknika honetan aktibazio funtzioaren deribatuak kalkultu behar dira, ondoren, katearen araua aplikatuz, hurrengo deribatuak kalkulatzeko, amaieran, pisuak pausu oro eguneratzeko. Zorionez, guk erabilitako software-an (Matlab-en) teknika hau automatikoki ejekutatzen da. Programadorea egin beharreko ataza bakarrak: sare neuronal mota aukeratzea, haren arkitektura eraikitzea eta geruzak aukeratzea izanda.



11.Irudia: MLP eta CNN arteko desberdintasunak.

Bi familia desberdinetan sailkatu ditzazkegu sare neuronalak (**11.irudia**): MLP *(MultiLayer Perceptron)*, funtzioak edo egoera bektorialak arazoa definitzen dituzten egoeratan gehien erabiltzen direnak, edo CNN *(Convolutional Neuron Network)*, normalean argazkiak sailkatzeko edo klasifikatzeko erabiltzen direnak.

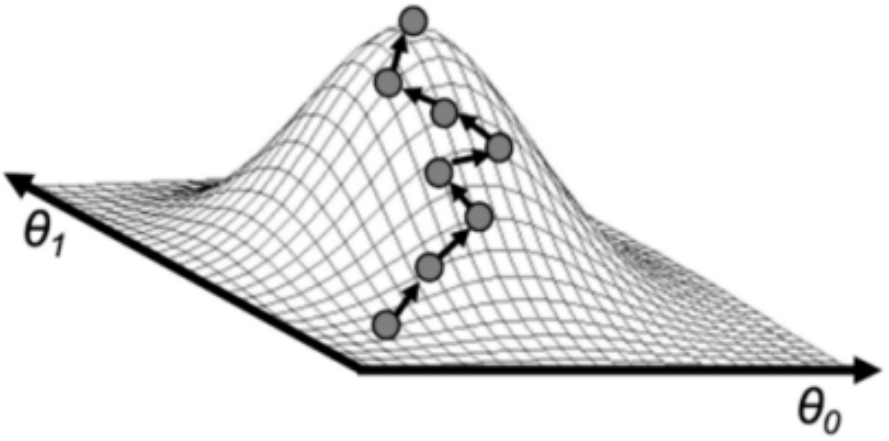
Sare neuronalek politika bat errepresentatu dezakete eta agentea izango da ekintzen probabilitate banaketa emango duena. Horretaz gain, balio funtzioa kalkulatzeko erabili daitezke. Gainera, entrenamendu algortimo batekin pisuen balioak aldatzen baditugu, agentearen konportamendua aldatuko dugu, honela egoera berrietara egokituz, politika optimoa lortu arte.

2.4. VALUE-BASED METODOA

Value-based metodoak **Q-funtzioa optimizatzen** dute, agentea aukera ditzazkeen ekintzen artean banaketa bat ezartzeko.

Estatu eta ekintza diskretuak dituen ingurune batean, Q funtzioa erabiliz, balio bat eman ahal diogu egoera bakoitzean eman daitezkeen egoera-ekintza pare guztiei (s,a). *Q-Learning* algoritmoa balio hoiek guztiekin taula bat eraikitzen eta optimizatzen du.

Ingurune jarraituetan, non, ekintza eta egoera balioak infinituak dira, horrelako taula bat eraikitzea ezinezkoa da. Kasu hauetan **DQN** (*Deep Q Network*) algoritmoa erabiltzen da, non, sare neuronalen bitartez ekintza-egoera pare guztien balioak aldi berean kalkulatu daitezke [8], honek banaka joatean baino askoz azkarragoa da [9]. Ondoren, antzezleak balio handiena duen ekintza aukeratzen du.



12.Irudia: Bi parametroekin sare neuronal baten ibilbidea, ekintza optimoen bila.

Sare neuronala optimizatzeko Q funtzioaren hurbilketa bat egiten da Bellman ekuazioaren bitartez [10].

2.5. POLICY SEARCH METODOA

Policy-search metodologian **politikak funtzio optimo bat bilatzen du** zuzenean, balio funtzioa kalkulatu gabe. Hau da, sare neuronala ez dugu ekintza-egoera pareen balioak emateko entrenatzeko, baizik eta, egoera partikular batean egin beharreko ekintza aukeratzeko.

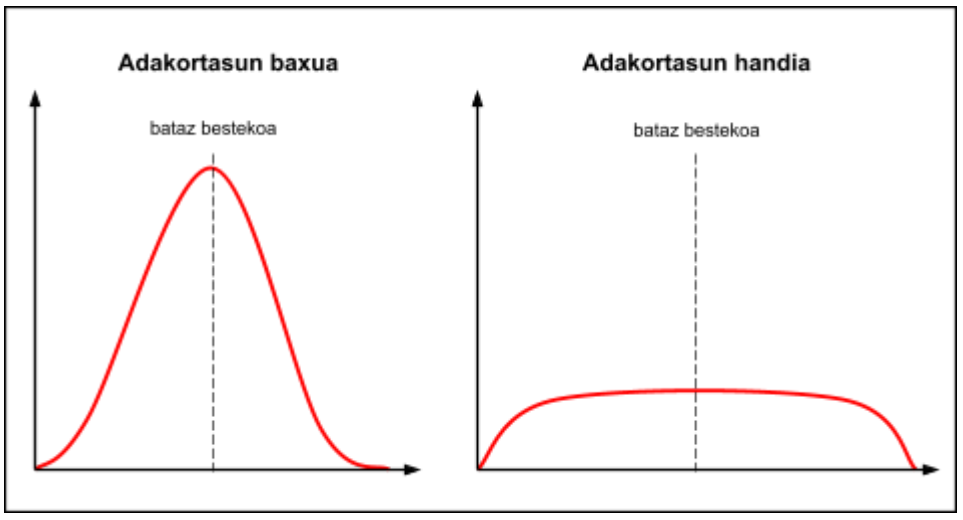
Policy-search metodoen barnean ***policy-gradient*** submetodoa da erabiltzen den gehiena. Honetan sare neuronalaren aldagaiak gradientearen maldarekiko eguneratzen dira (**12.irudia**). Gradientea *time step* bakoitzean, aldagaiak aldatzeko hartu beharreko noranzkoa (malda handieneko norabidea) ematen digu .

2.6. ACTOR-CRITIC METODOA

Actor-critic metodologia, *value-based* eta *policy-seach* arteko nahasketa da.

Policy-gradient metodoen arazo larrienetariko bat egonkortasuna da, hau da, espero ditugun probabilitateak eta sariaren balioak aldakortasun handia dute.

“Aldakoratasuna” termino estadistiko bat da, probabilitatearen distribuzioa neurtzen duena. Esaterako, distribuzio arrunt batean, **13.irudian** aldakortasunaren garrantzia isladatzen duten bi grafika ikus ditzazkegu:



13.Irudia: Aldarkotasunaren irudikapena, distribuzio arrunt batean.

Balio funtzioan gradientea **7.ekuazioaren** bitartez kalkultzen da:

$$\nabla_{\theta} U(\theta) \leftarrow \sum_{t=0}^{H} \nabla log \pi_{\theta}(a_t|s_t) G_t \quad (7)$$

Imagina dezagun 3 kapitulu desberdin, non, G1 eta G2 balio positibo oso antzekoak dituzte, baina G3 balio negatibo desberdina hartzen du. Ondorioz, *"reward"*-a t=1-n eta t=2-n balio positibo bat izango du eta t=3-n aldiz negatiboa. Gradientea 3 kapituluak konsideratzen baditu, politika aldatzerakoan 3.kapituluan hartutako ekintzatik asko urrrunduko du eta 1 eta 2.kapituluan hartutako ekintzetara hurbildu. Halere, imagina dezagun orain 3 kapituluak balio positiboak dituztela, baina G1 eta G2 balio oso handiak izango dituzte eta G3 aldiz, oso txikia (zerorekiko oso hurbila). Kasu honetan gradientea politika t=1 eta t=2 hartutako ekintzei hurbilduko du, t=3 hartutakoa alde batera utziz.

Ondorioz, gradientearen menpekotasuna balio funtzioaren erantzunen bata-besteraekiko oso handia da. Efektu hau txikitzeko ***"baseline"*** kontzeptua atera zen. *Baseline-a* espero ditugun balio funtzioen batabesteko da. Baina nola erabili hau *actor-critic* metodologian?

Actor critic metodologian *baseline*-a ez dago espero ditugun sarien erantzunen menpe, baizik eta, inguruaren egoeraren menpe dago. Hau ulertzeko, DQN algoritmoa hartu behar dugu, non, egoera-ekintza pare bakoitza balio Q(a,s) bat zuen.

Egoera balio intrinseko bat badu, ekintzaren balioa **8.ekuazioaren** itxura hartzen du.

$$Q(a, s) = V(s) + A(s, a) \quad (8)$$

Kasu honetan, **V(s)** ***baseline*** **balioa** izango da eta **A(a,s) transizioaren balio** emango digu.

V(s)-ren balioa hasiera batean ezezaguna da, halere DQN algoritmoan bezala sare neuronal baten bitartez kalkulatu dezakegu. Sare neuronal hau DQN algoritmoak Bellman ekuazioa sare neuronal-en pisuak kalkulatzeko erabiltzen zuen era berean entrenatzen da. Sare neuronal honi "**Kritikoa"** deitzen diogu. Laburbilduz, "Kritikoak" ingurua aurkitzen den egoerari balio bat ematen dio, V(s).

Kritikoarekin batera, ekintzak aukeratzen dituen sare neuroalaren pisuen balioen aldaketa optimiza dezakegu, pisu hauek gure **politika (π)** sortuko dute. Ekintzak aukeratzen dituen bigarren sare neuronal honi **Antzezlea** deitzen diogu.

Praktikan sare neuronal hauek batera egiten dute lan (ikus **1.algoritmoa**); Izan ere, bi sareak duten sarrera elementua bera da, hasieran aipatutako behaketa-espazioa.

1.Algoritmoa: Actor-Critic

```
1.  Hasieratu programa.Sare neuronalaren balioak (pisuak) ausazkoak.
2.  while not done
3.        Play N steps in the environment with π0 saving epoch of (st,at,rt).
4.        if (done)
5.              R = 0
6.        else
7.              R=Vtheta(st)
8.        end
9.        Update backwards:
10.       for i = t-1…tstart
11.             R ← ri + γR
12.             Accumulate policy gradients.
13.             Accumulate value gradients.
14.             Update networks with gradients.
15.       end
16. end
17. return π
```

Actor-critic metodologia oinarritzat hartzen duten sistema asko daude, garrantzitsuenak DDPG[11], SAC[12] eta PPO[13] dira.

3. DETERMINISTIC POLICY GRADIENT.

Algoritmo honetan kritiko eta antzezle bat izatearen ideia hartzen da eta *off-policy* metodologiarekin konbinatzen da.

Antzezlearen funtzioa egoera bat ekintza batean bilakatzea izango da, μ(s). Funtzio hau kritioaren barnean sartu dezakegu, Q-balio funtzioa honela gera dezan Q(s,μ(s)).

Sare neuronalak, funtzio matematiko moduan erabiliko ditugu.

Ondorioz, orain Q-balioa bi sare neuronalen (antzezle, θμ parametroak, eta kritiko, θQ parametroak) baitan geratuko da. Gure helburua, bi hauen pisuak optimizatzea da balio hau maximizatzeko. Horretarako, gure politkaren gradientea lortu behar dugu, pisuak ze norabidean eguneratzeko jakiteko.

[5] artikuluan David Silver frogatu zuen moduan, politika estokastiko baten gradientea politika deterministiko baten gradientearen berdina da. Beraz, politika optimizatzeko soilik kalkulatu behar dugu Q(s,μ(s)) funtzioaren gradientea **9.ekuazioan** ikus dezakegun moduan.

$$\nabla_s Q(s, \mu(s)) = \nabla_s Q(s, \mu(s)) \nabla_{\theta_\mu} \mu(s) \tag{9}$$

Deterministic Policy Gradients ideiaren barnean bi sailkapen egin ditzazkegu: DDPG (*Deep Deterministic Poicy Gradient*) eta A2C.

A2C-n kritikoa baseline balioa erabiltzen du eta honi egonkortasuna hobetzeko jasotako saria geitzen zaio.

DDPG-n, berriz, politika deterministikoa da. Q(s,μ(s)) funtzioaren gradienteak kritikoa kalkulatzen du, antzezleak aukeratutako ekintzen bitartez. Era honetan, sistema guztia gradientearen optimizazioarekin batera aldatzen da.

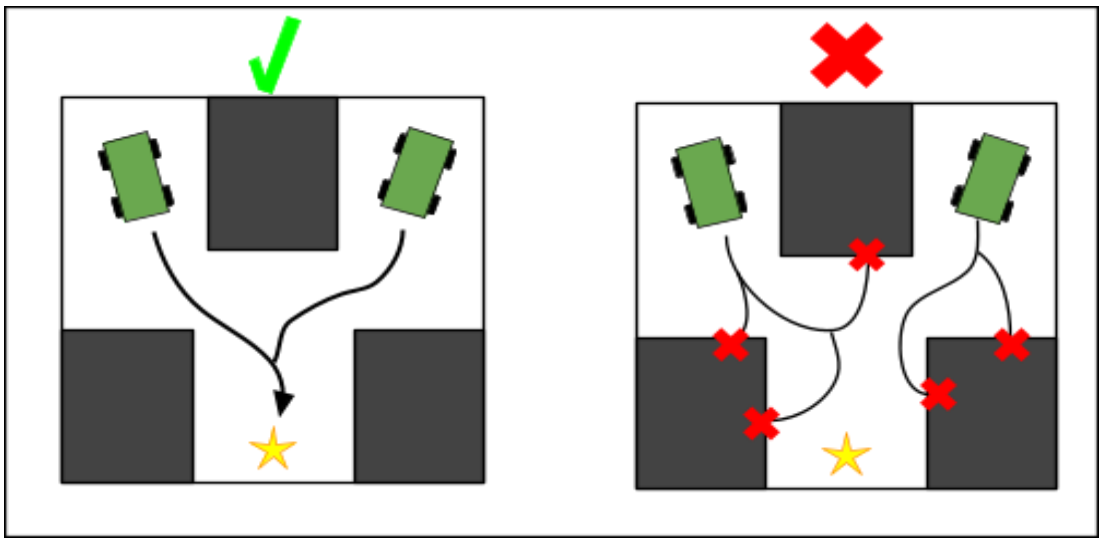
3.1. ADIBIDEA:

DDPG agente eta ingurune bat sortuko dugu. DDPG agentea soilik ekintza jarraiak jasaten ditu. Ekintza jarraiek ez dute balio limiterik, hartzen ditugun bi balioen artean balio infinitoak egon daitezke. (Kodigoa: A Eranskina)

3.1.1. Arazoa.

Biltegi batean bere baitan mugitzeko ahalmea duen robota sortuko dugu. Horretarako, biltegiaren simulazio bat sortu beharko dugu. Robota mugitzerakoan biltegi guztia ikus dezagun, simulazio guztiak goiko ikuspegi batekin emango da, **14.irudian** ikusten den moduan.

Gure helburua, robota ausazko hasiera posizio batetik amaiera posizioa batera heltzea izango da, ingurunearekin kolisiorik egin gabe eta denbora tarte murriztu batean, **14.irudian** ikusten den moduan.



14.Irudia: Biltegi robota duen helburuaren irudikapena.

3.1.2.- Ingurua.

Behaketa-espazioan, 6 aldagai egongo dira. Aldagaiek balio limiterik ez dutenez, behaketa espazioan sartuko dugun gauza bakarra aldagai kantitatea izango da (ikus **2.algoritmoa**). Hauek dira sortuko ditugun aldagaiak:

- **x** eta **y** robotaren posizioa emango digute.
- **sin(θ)** eta **cos(θ),** robotaren orientazioa emango digute.
- **v**, robotaren abiadura lineala emango digu.
- **ω**, robotaren abiadura angeluarra emango digu.

Agenteak aukeratu ditzazkeen aldagaiak (ekintzak) indar normalizatuak dira. Horregatik, lehen aipatu dugun moduan, Matlab-en ez dugu aldagaien informaziorik sartuko, soilik kantitatea. Halere, indarra normalizatuak egongo direnez, goiko eta beheko limiteak ezarriko ditugu (ikus **3.algoritmoa**).

Hauek izango dira gure indarrak:

- Biraketa indarra: **Frot** deituko duguna.
- Impultso indarra: **Ftrans** deituko duguna

2.Algoritmoa: Inguruaren sorketa, ekintza jarraituentzako.

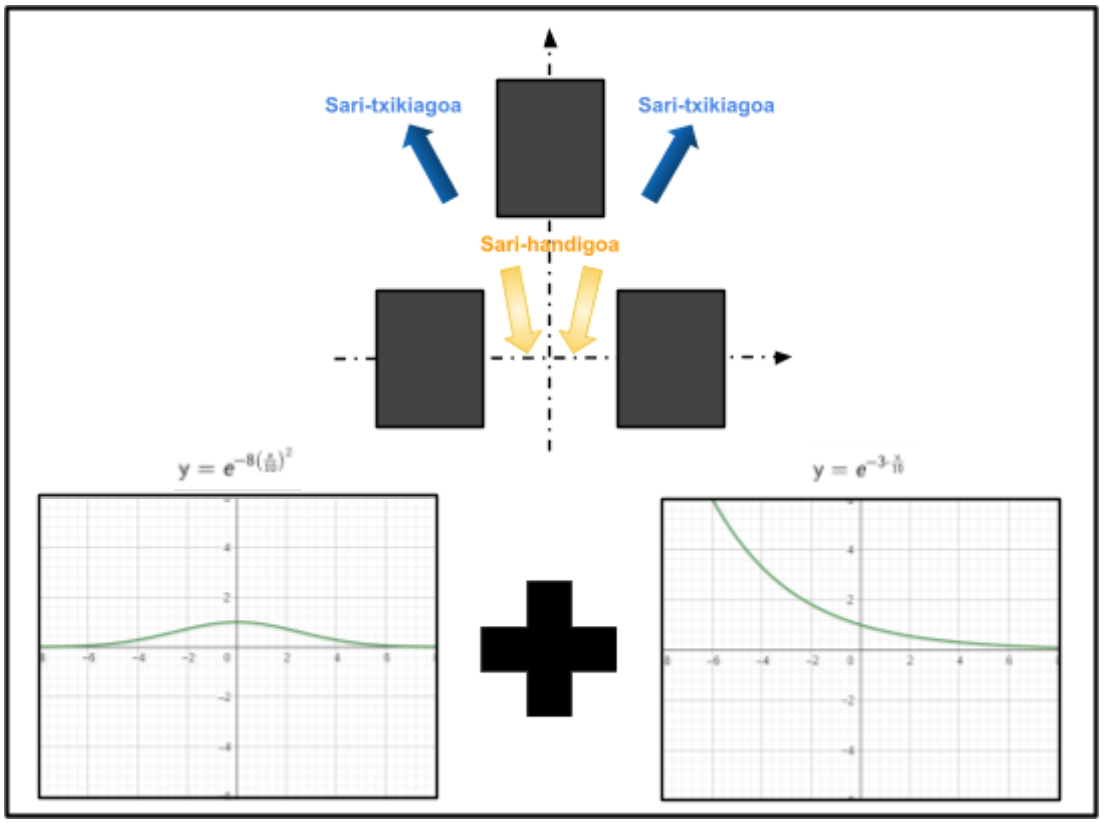
1. **Behaketa eta aukeren balioak zehaztu:**
2. Behaketak ← [6 1]. Behaketak 6 izango dira.
3. Behaketa informazioa ← Behaketen informazioa sartu. [rlNumericSpect() [14]]
4. Aukerak/Ekintzak ← [2 1]. Soilik bi indar kontrolatuko ditugu.
5. Ekintza informazioa ← Aukeren/Ekintzen informazioa sartu. [rlNumericSpect() [14]]
6. **Ingurua sortu:**
7. slEnv ← [rlSimulinkEnv() [15]] (Gogoratu simulink moduluarekin erlazionatzea)
8. slEnv ← "randomstart" funtzioa Reset.Fcn funtzioa ordezkatu du.

3.Algoritmoa: Reset funtzioa.

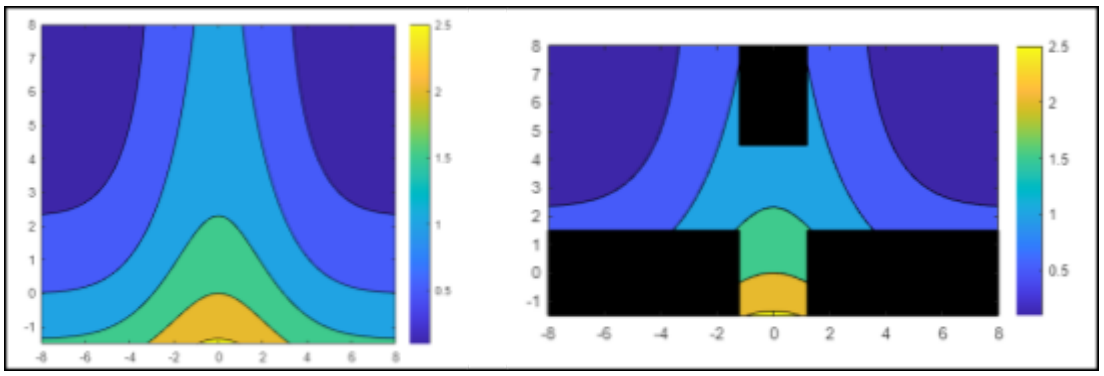
1. **function** randomstart (in)
2. in← Hasierako posizio x0, auzakoa izango da. (setVariable() [16])
3. in← Hasierako posizio x0, auzakoa izango da.
4. in← Hasierako orientazioa θ, auzazkoa izango da..
5. in← Hasierako abiadura lineala v0, auzazkoa izango da.
6. in← Hasierako abiadura angeluarra w0, auzazkoa izango da.
7. **return** in

3.1.3. Sari funtzioa (Reward function).

Sari funtzioak robota, **15 eta 16.irudietan** ikusten den moduan, **14.irudian** dagoen izarrera mugitzera animatzen du. Puntu honetan, gure sistema metrikoaren hasiera ((x,y)=(0,0)) jarriko dugu.



15.Irudia: Reward funtzioen irudikapena eta osatzen duten funtzioen grafikak.



16.Irudia: Sari funtzioa inguruan duen aldaketaren azalpen grafikoa.

Sari funtzioa simulink-en ereduaren barnean kalkulatzen da, 5 sarrera izango ditu: posizioa x-ardatzean, posizioa y-ardatzean, abiadura angeluarra, helburua lortu duen adierazten duen aldagai bat eta kolisio bat edo errore bat egon den adierazten duen aldagaia (**9.ekuazioa**).

$$r(x, y, \Omega, madeit, collided) = 0.05 * e^{-8*\left(\frac{x}{10}\right)^2} + 0.06 * e^{-3*\frac{y}{10}} - 0.14 - 0.001 * \Omega^2 - 5 * madeit - 2 * collided \quad (9)$$

Aurrerago, sari honek ez zuela robota mugimendu konstante bat egitera bultzatzen ikusi dugu. Sari funtzio honekin, robota guztiz azeleratzeko eta ondoren gelditzeko joera hartzen du. Ondorioz, behekaldean ikus dezakezun ekuazioa (**10.ekuazioa**) *reward* funtzioaren barnean sartuko dugu, mugimendu konstantea bultzatzeko.

$$0.01 * [F_{trans}^2 + F_{rot}^2] \quad (10)$$

3.1.4. Agentea.

DDPG agenteek "gradiente politika deterministiko" bat jarraitzen dute sare neuronalen bitartez. Barruan antzezle deterministiko bat eta Q-balioa kalkulatzen duen kritiko bat dute.

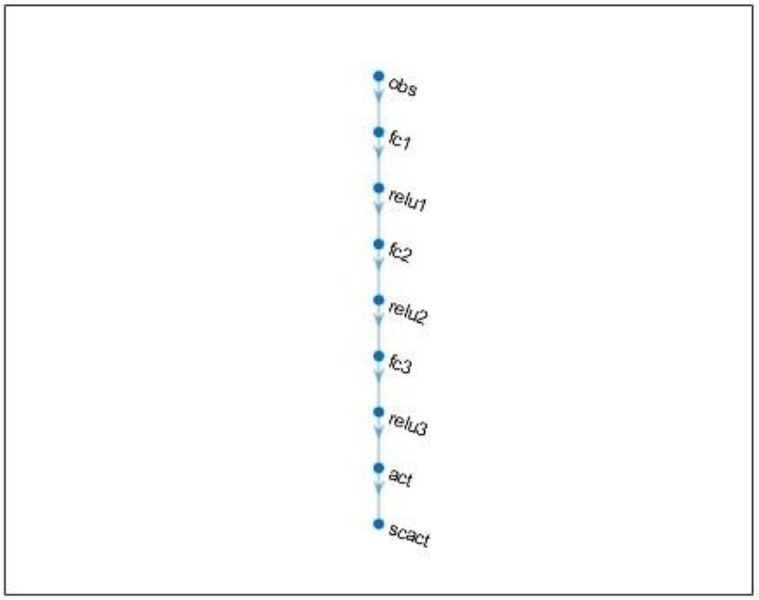
3.1.4.1. Antzezlea:

Sortuko dugun antzezlea, ya da esan dugun moduan deterministikoa izango da, hau da, aukera ditzekeen ekintza multzoa infinitua da. Eraikiko dugun sare neuronala, hurrengo geruzak izango ditu (**17.irudia**) [17]:

- Sarrera geruza *(Input Layer)*: Datuak jasoko ditu. Jasotako datu kantitatea izango dituen neurona kantitatearen berdina izango da. Sei neurona izango ditu.
- Lehenengo konexio geruza *(Fully Connected Layer)*: Frogak egin ondoren neurona kantitate optimoa 100 neurona direla ikusi dugu. Konexio geruza guztiak, barnean dituen pisuekin, gure robotak egingo dituen indarrak kontrolatuko dute.
- Lehenengo ReLu geruza *(ReLu Layer)*: Balio negatiboak ezabatzen ditu, funtzionamendua azkarrago izateko.
- Bigarren konexio geruza *(Fully Connected Layer)*: 100 neurona.
- Bigarren ReLu geruza *(ReLu Layer)*.
- Hirugarren konexio geruza *(Fully Connected Layer)*: 100 neurona.
- Hirugarren ReLu geruza *(ReLu Layer)*.

- Azkenengo konexio geruza *(Fully Connected Layer)*: Bi neurona. Neurona kantitatea ekintzak izango dituen aldagai kantitatearen berdina izango dira. Gure kasuan bi indar kontrolatuko ditugu.
- Amaierako geruza *(tanh layer)*: Balioak normalizatzen ditu 1 eta -1 artean.

Entrenamenduan, aukeratutako ekintzei soinu bat sartuko diegu. Honek agenteak aukera gehio arakatzea ahalbideratzen du.



17.Irudia: Antzezlearen sare neuronalaren irudikapena.

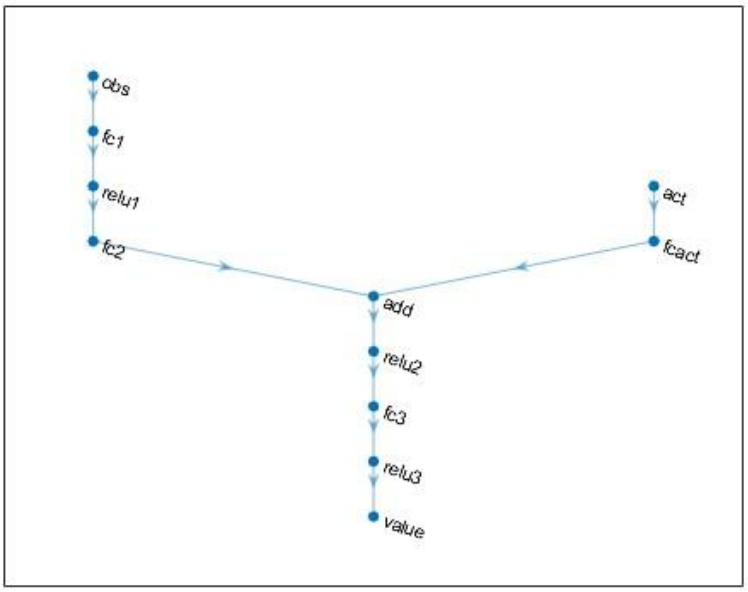
3.1.4.2- Kritikoa:

Q-balio funtzioa kalkulatzen duen kritiko bat eraikiko dugu. Sarrera bi ilara desberdinetan hasiko da, amaieran, ilara bakarrean bukatuz (**18.irudia**). Honen arrazoia kritikoa bi sarrera desberdin izatea da: behaketak eta ekintzak.

- Lehenengo sarrera ilara (behaketak):
    - Sarrera geruza *(Input Layer)*: Behaketa datuak jasoko ditu. Jasotako datu kantitatea izango dituen neurona kantitatearen berdina izango da. 6 behaketa ditugunez 6 neurona izango ditugu.
    - Lehenengo konexio geruza *(Fully Connected Layer)*: Frogak egin ondoren neurona kantitate optimoa 100 neurona dela ikusi dugu.

- ReLu geruza *(ReLu Layer)*: Balio negatiboak ezabatzen ditu.
- Bigarren konexio geruza *(Fully Connected Layer)*: 100 neuronekin.

- Bigarren sarrera ilara (ekintzak):
  - Sarrera geruza *(Input Layer)*: Ekintza datuak jasoko ditu. Jasotako datu kantitatea izango dituen neurona kantitatearen berdina izango da. 2 ekintza ditugunez 2 neurona izango ditugu.
  - Lehenengo konexio geruza *(Fully Connected Layer)*: Frogak egin ondoren neurona kantitate optimoa 100 neurona direla ikusi dugu.
- Irteera ilera bakarra:
  - Batuketa geruza *(Additon Layer)*: Bi ilarak batzen ditu.
  - ReLu geruza *(ReLu Layer)*: Balio negatiboak ezabatzen ditu.
  - Azkenengo konexio geruza *(Fully Connected Layer)*: Neurona bakarra. Hemendik Q-balio funtzioa antzezleari eramango zaio.



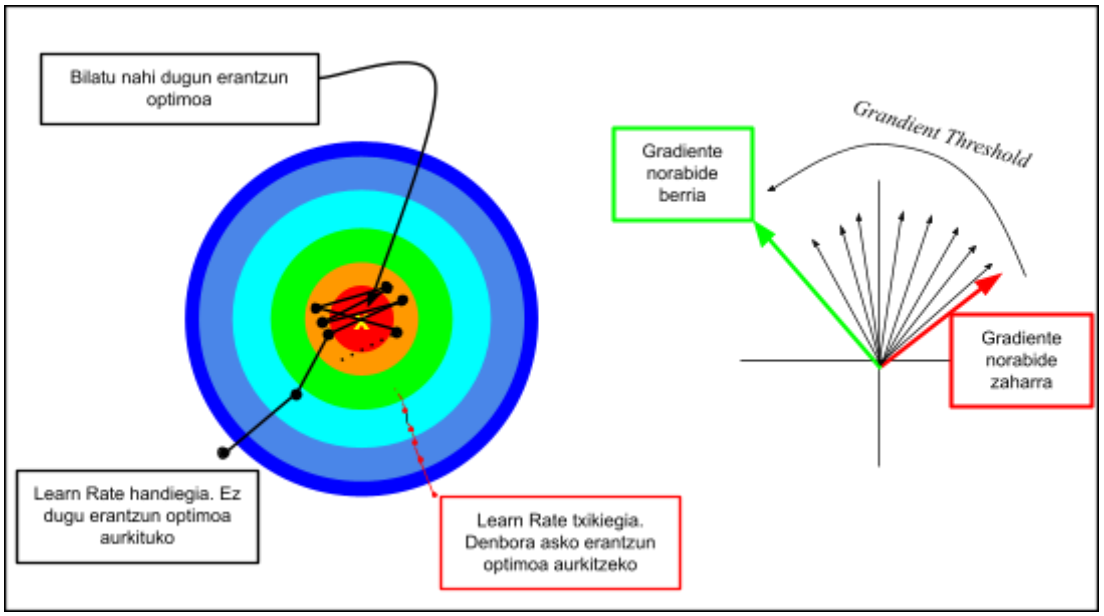
18.Irudia: Kritikoaren sare neuronalaren irudikapena.

3.1.5. Entrenamendua.

Entrenamendu bat hasterakoan, garrantzitsuena aukeren balioak erabakitzea da. Aukerak bai kritiko, bai antzezle, bai agentean eta entrenamenduan zehaztuko ditugu. Guk ataza honetan hurrengo aukerak zehaztuko ditugu, halere, ikus [18] aukera guztiak begirazteko.

Antzezlea eta kritikoarentzako aukera bedinak ezarriko ditugu.

- *"Learn Rate"* edo Ikasketa Abiadura: Antzezlearen eta kritikoaren aldagaiak zenbat aldatzen diren ezartzen du. Imagina dezagun gure ikasketa **12.irudia** irudikatzen duela, gure erantzuna optimoa mendiaren goikaldean egongo da eta abiadura ikasketa egingo ditugun pausuen distantzia izango da. Oso txikia bada, ikasketa denbora asko iraungo du erantzun optimoa aurkitu arte, baina oso handia bada ikasketa erantzun suboptimo batean gera daiteke edo erantzuna inoiz ez aurkitu (**19.irudia**). Guk "5e-5" hautatu dugu.
- *"Grandient Threshold"*: Interakzio bakoitzean gradientearen norabide berrian zenbat mugitzen garen adierazten du (**19.Irudia**). Guk 10-era aldatu dugu.



19.Irudia: Kritiko eta antzezlearen aukeren irudikapena.

Aurrerago agentearen aukerak hautatuko ditugu [19]:

- *"Sample Time"*: Interakzioen arteko denbota ezartzen du. Guk 0.25 segundutara ezarri dugu.
- *"Experience Buffer Length"*: Agenteak experientzian oinarritutako ikasketa egiten duenez, zenbat experientzia gordeko ditugun adierazi beharko dugu. Zenbat eta, handiagoa, orduan eta, ikasketa hobeagoa izango dugu, baina konputazio-betekizun handiagoak beharko ditugu. Gure gordetako experientziak 1000000 izango dira.
- *"Mini Batch Size"*: Gordetako experientzia guztietatik interakzio bakoitzean ikasteko zenbat hartzen ditugun adierazten du. Guk 128 experientzietatik ikasiko dugu interakzio bakoitzean.

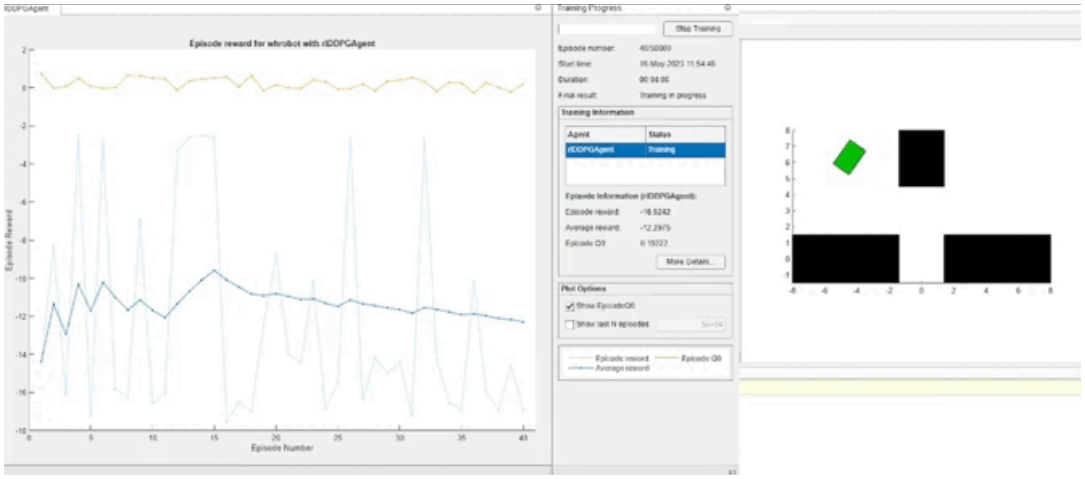
Horretaz gain, entrenamendu aukerak ere zehaztuko ditugu. Hona hemen guk hautatu ditugunak [20]:

- "*Maximum Steps Per Episode"*: Kapitulu batean agentea eta inguruaren artean zenbat "Steps" edo interakzio egongo diren zehazten du. Honekin kapitulu baten simulazio denbora maximoa ere atera dezakegu. Guk aldagai hau 120 interakzioetan jarri dugu. Ondorioz, gehienez kapitulu bat 30s (120 * 0.25s) iraungo ditu.
- *"Max Episodes"*: Kapitulu kantitate maximoa zehazten du. Gure kasuan gehienez 50000 kapitulu izango ditugu.
- *"Score Averaging Window Length"*: Hurrengo aukeran ikusi dezakegun moduan, gure agenteak bataz besteko sari bat lortzerakoan gordeko da. Bataz besteko hori azkenengo zenbat kapituluekin kalkultzen den zehazten du aukera hau. Gure kasuan azkenengo 50 kapituluekin kalkulatuko da.
- *"Stop Training Criteria , Average Reward, Stop Training Value"*: Agente bat gordetzeko da bataz besteko saria 2 izan beharko da.

3.1.6. Entrenamenduaren erantzuna.

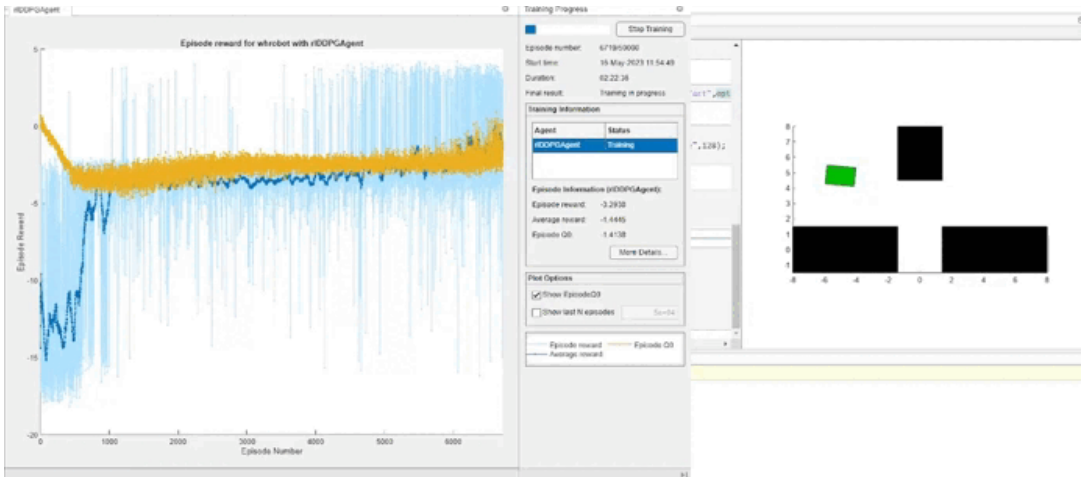
Entrenamendua hurrengo bi iruditan ikus daitekeen moduan eman da.

20.Irudian, hasieran, gure robota ez du biltegitik bide logiko bat jarraitzen. Lehenengo aukerak normalean ausazkoak direnez, hasierako simulazio guztiak huts egiten dute bi arrazoiengatik: robota biltegiko beste objektu batekin talka egiten du edo robota bueltaka dago une oro, helburua lortu gabe simulazio denbora bukatu delako. Horregatik, hasierako saria *(reward-a)* balio oso txikiak izan ditu.



20.Irudia: Entrenamendua hasieran.

21.Irudian, aldiz, agentea entrenamenduan denbora zintzo bat egon da. Ikusi dezakegun moduan, honek robota ekintza logikoagoak hartzea ekarri du. Ya ia simulazio guztietan helburua lortzen du.

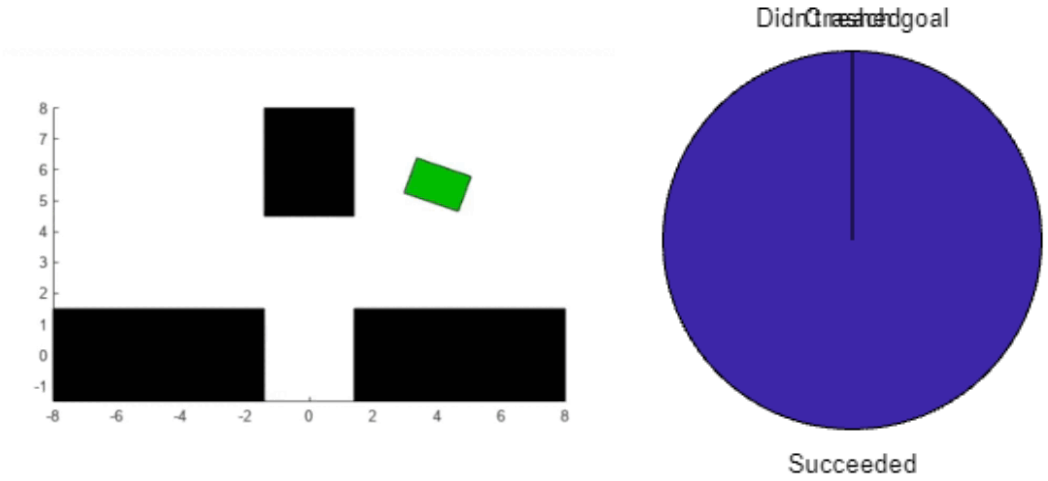


21.Irudia: Entrenamendua amaieran.

Entrenamendua agenteak guk nahi genuen balioak lortu dituenean gode dugu, "*saved_agent"* izenarekin. Honekin komprobaketak egingo ditugu.

3.1.7. Komprobaketa.

Gordetako agentearekin, simulink eredua erabiliz, 20 simulazio egin ditugu. **23.Irudian** ikus dezakegun moduan, robota simulazio guztietan helburua lortzen du.



23.Irudia: Konprobaketaren erantzuna entrenatutako agente batekin.

Hau honela izanda, segurtasunarekin esan dezakegu DDPG agente bat sortu eta entrenatu dugula, bere helburua lortzeko.

4. PROXIMAL POLICY OPTIMIZATION

PPO metodoa John Schulman eta beste egile batzuk 2017 sortu zuten **TRPO algoritmoaren** simplifikazio bat da [13].

A2C metodologian du oinarria, baina gradientearen hurbilketa egiteko erabiltzen den formula ez da bedina. Ekintza probabilitate distribuzioaren gradientea erabili ordez, politika berri eta zaharraren arteko ratioaren gradientea erabiltzen du.

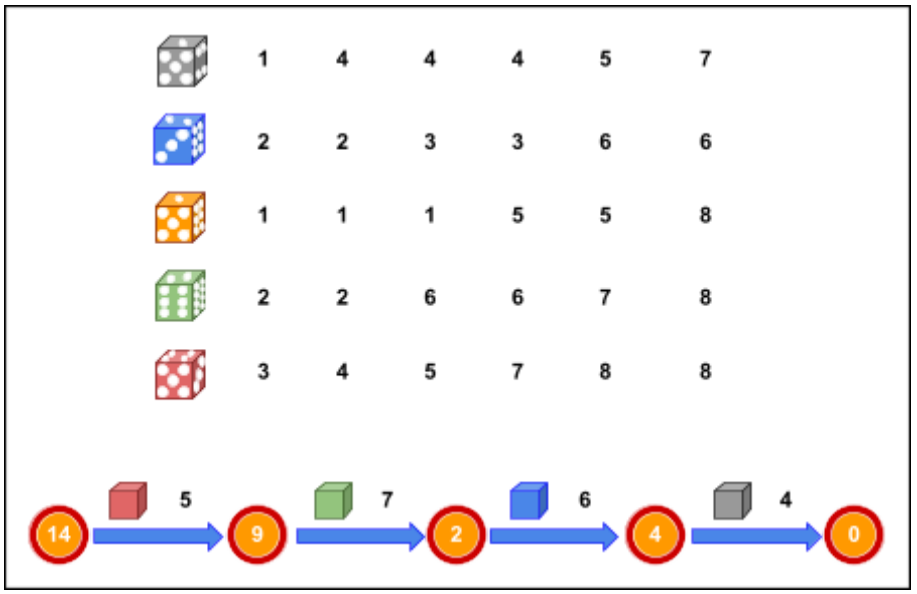
Horretaz gain, PPO *"experience buffer"*-a (gordetako experientzia kantitatea) erabiltzen du ikasketan.

4.1. ADIBIDEA:

PPO metodoaren adibide bat ikusteko, jolas simple batera jolastuko dugu. PPO ekintza diskretu eta jarraiak jasaten dituenez, daduaren jolaserako algoritmo egokia da. (Kodigoa: B Eranskina)

4.1.1. Arazoa.

Jolasa sinplea da. Bost dadu desberdin izango ditugu, bakoitza zenbaki eta kolore desberdinekin (**24.irudia**). Hasieran, balio bat hartuko dugu, gure kasuan 20 izango da. Dadu bat hartu eta lehenengo aldiz botako dugu. Ateratako zenbakia, genuen balioari kenduko diogu. Hau errepikatuz 0-ra ahalik eta azkarren iritsi arte.



24.Irudia: Daduen jokoaren irudikapena.

4.1.2. Ingurua.

Inguruan egoera multzoa finitua da; Izan ere, gure egoeraren balioa beti 1 eta 20 artean egongo da. Orduan, gure egoeren informazioa 0tik 20ra doan bektore moduan adieraziko dugu (ikus **4.algoritmoa**).

Agenteak hartu ditzazkeen aukerak ere mugatuak (diskretuak) dira. Ditugun daduen artean aukeratu beharko dugu. Gure aukeren informazioa 1tik 5ra doan bektore bat izango da (ikus **4.algoritmoa**).

4.Algoritmoa: Inguruaren sorketa, ekintza diskretuetarako.

```
1.  Daduen eta aukeren balioak zehaztu:
2.      d ←Daduen informazioa. 5*6 Matrizea.  [1 1 1 5 5 8;2 2 3 3 6 6;1 4 4 4 5 7;2 2 6 6 7 8;3 4 5 7 8 8]
3.      Egoerak ← [0:20]
4.      Aukerak/Ekintzak ← [1:5]
5.      Behaketa informazioa ← Egoeren informazioa sartu. [ rlFiniteSetSpect() [21]]
6.      Ekintza informazioa ← Aukeren/Ekintzen informazioa sartu. [ rlFiniteSetSpect() [21]]
7.  Ingurua sortu
8.      slEnv ← [rlSimulinkEnv() [15]]  (Gogoratu simulink modulua)
9.      slEnv ← "randomstart" funtzioa Reset.Fcn funtzioa ordezkatu.
```

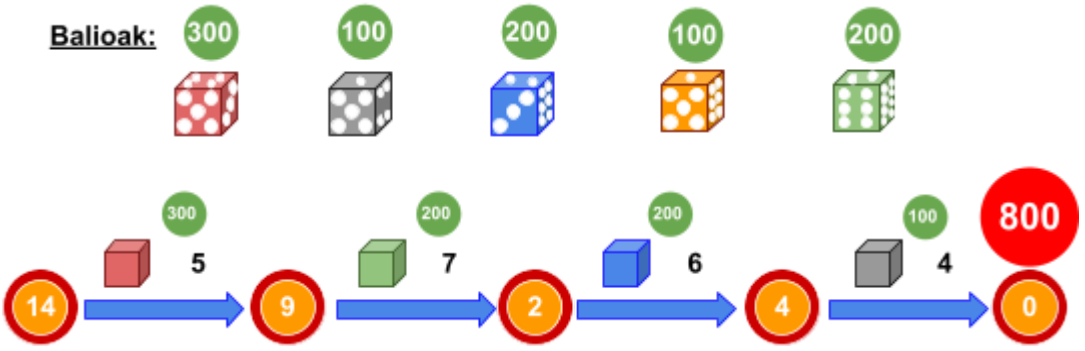
5.Algoritmoa: Reset funtzioa.

```
1.  function randomstart (siminput)
2.      siminput← Hasierako posizio "state", 20 izango da (setVariable() [16] ).
3.      siminput← Hasierako posizio balioa, 0 izango da.
4.  return siminput
```

4.1.3. Sari funtzioa (Reward function).

Kasu honetan zaila da sari funtzioa imaginatzea; Azken batean, jokoaren helburua soilik 0-ra iristea da. *"Reward"* funtzioa idazteko aldaketa txiki bat egingo diogu gure jokoari.

Orain, dadu bakoitza puntuaketa bat izango du. Puntuaketa hau daduak botatzen ditugun bakoitzean gehitzen joango da. Orduan, helburua orain ez da soilik 0-ra iristea izango, baita ahalik eta puntu gutxien ateratzea ere. Funtzioari kasu honetan "saria" deitu beharrean "kostua" deituko diogu; Izan ere, zenbat eta, balio handiagoa izan, orduan eta, txarragoa izango da gure partida (**25.irudia**).



25.Irudia: Daduen jokoen irudikapena, sari funtzioaren barneraketarekin.

6.Algoritmoa: Sari funtzioa, dadu jolasean.

```
1.  function (egoera,balioTotala,daduZenbakia,daduBalioa)
2.  balioTotala=balioTotala - daduBalioa. ← Jokoaren puntuazioa kalkulatu.
3.  Reward=balioTotala.
4.  egoera = egoera - daduZenbakia. ← Egoera eguneratu
5.      if egoera < 0
6.          egoera = (-1)*egoera ← Jokoa ez da bukatzen 0-ra iritsi arte.
7.          isDone = 0.
8.      elseif egoera == 0
9.          Reward = Reward+10000 ← Jokoa bukatu da eta helburua lortu da.
10.         isDone = 1.
11.     else
12.         isDone = 0.
13.     end
14.   Behaketa = egoera.
15. return Reward,isDone,Behaketa.
```

4.1.4. Agentea.

Bi sare neuronalak sortuko ditugu agentearen barnean: kritikoa eta antzezlea.

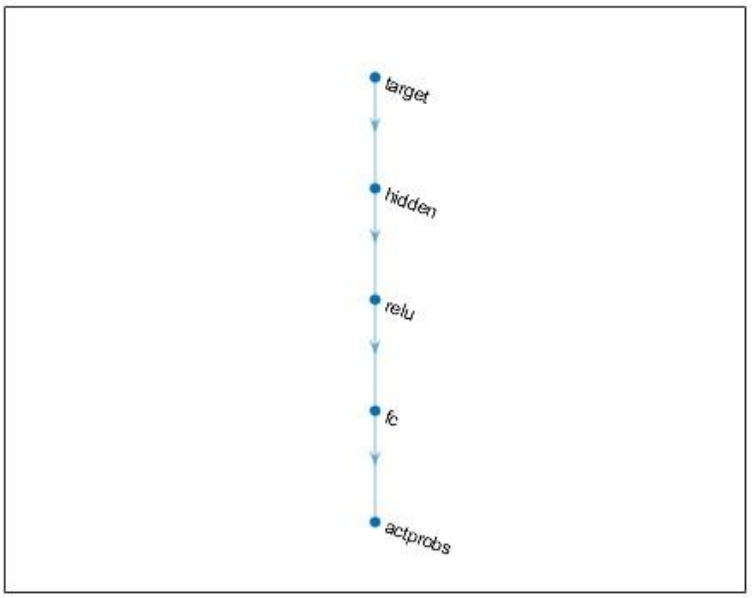
Halere, hainbat froga ondoren, esan beharra dago, sortutako aplikazioa gehiegi izan daitekela guk proposatutako jolasan. Hau konputazio ahalmen handiak behar ditu eta gure jolasa soilik Q-taula aztertzen duen kritiko batekin konpondu izan genezake.

4.1.4.1. Antzezlea:

Sortuko dugun antzezlea, estokastikoa izango da, hau da, aukera ditzekeen ekintza multzoa finitua izango da (soilik 5 dadu desberdin ditugu). Eraikiko dugun sare neuronala, hurrengo geruzak izango ditu (**26.irudia**) [17]:

- Sarrera geruza *(Input Layer)*: Datuak jasoko ditu. Jasotako datu kantitatea izango dituen neurona kantitatearen berdina izango da. Neurona bakarra izango du.
- Lehenengo konexio geruza *(Fully Connected Layer)*: Frogak egin ondoren neurona kantitate optimoa 20 neurona direla ikusi dugu. Hau aukeraketa funtzioa kontrolatzen du, bere pisuen bitartez.
- ReLu geruza *(ReLu Layer)*: Balio negatiboak ezabatzen ditu, funtzionamendua azkarrago izateko.
- Bigarren konexio geruza *(Fully Connected Layer)*: Azkenengo konexio geruza denez eta ditugun aukerak finitoak direnez, neurona kantitatea aukera kantitatearen berdina izan beharko da, 5.
- Irteera geruza *(softmax Layer)*. Aurreko geruzatik balio handiena, aukera hoberena, hartzen duen geruza da, ondoren, ingurunera transmitituz.

Entrenamenduaren hasieran dadu aukera auzaskoa dela aipatzea merezi du, honela gure agentea ahal dituen posibilitate gehienak aztertzen ditu.

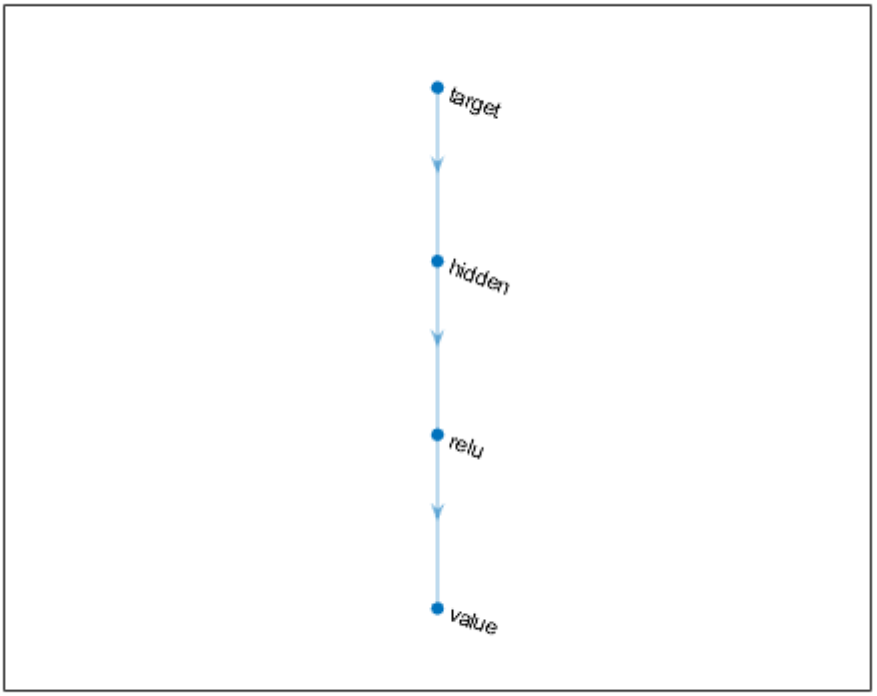


26.Irudia: Antzezlearen sare neuronalaren irudikapena.

4.1.4.2. Kritikoa:

V-balio funtzioa kalkulatzen duen kritiko bat eraikiko dugu. Eraikiko dugun sare neuronala, hurrengo geruzak izango ditu (**27.irudia**) [17]:

- Sarrera geruza *(Input Layer)*: Datuak jasoko ditu. Jasotako datu kantitatea izango dituen neurona kantitatearen berdina izango da.
- Lehenengo konexio geruza *(Fully Connected Layer)*: Frogak egin ondoren neurona kantitate optimoa 20 neurona direla ikusi dugu. Hau Q-balio funtzioaren hurbilketa kontrolatzen de, bere pisuen bitartez.
- ReLu geruza *(ReLu Layer)*: Balio negatiboak ezabatzen ditu.
- Bigarren konexio geruza/Irteera geruza *(Fully Connected Layer)*: Egoeraren balioa da gure irteera bakarra, ondorioz, neurona bakarra izango du.



27.Irudia: Kritikoaren sare neuronalaren irudikapena.

4.1.5. Entrenamendua.

Antzezle eta kritikoaren barnean bi aukera aldatuko ditugu, bestetan balioa lehenetsiak utziko ditugu. “Default” balio ikusteko joan [22]-ra.

Antzezlearen aukerak:

- *“Learn Rate”* edo Ikasketa Abiadura: Guk “2e-04”-ra zehaztuko dugu (ikus **19.irudia**).
- *“Grandient Threshold”*: Interakzio bakoitzean gradientearen norabide berrian zenbat mugitzen garen adierazten du (**19.irudia**). Guk 1-era aldatuko dugu.

Kritikoaren aukerak:

- *“Learn Rate”* edo Ikasketa Abiadura: Guk “1e-03”-ra zehaztuko dugu.
- *“Grandient Threshold”*: Guk 1-era aldatuko dugu.

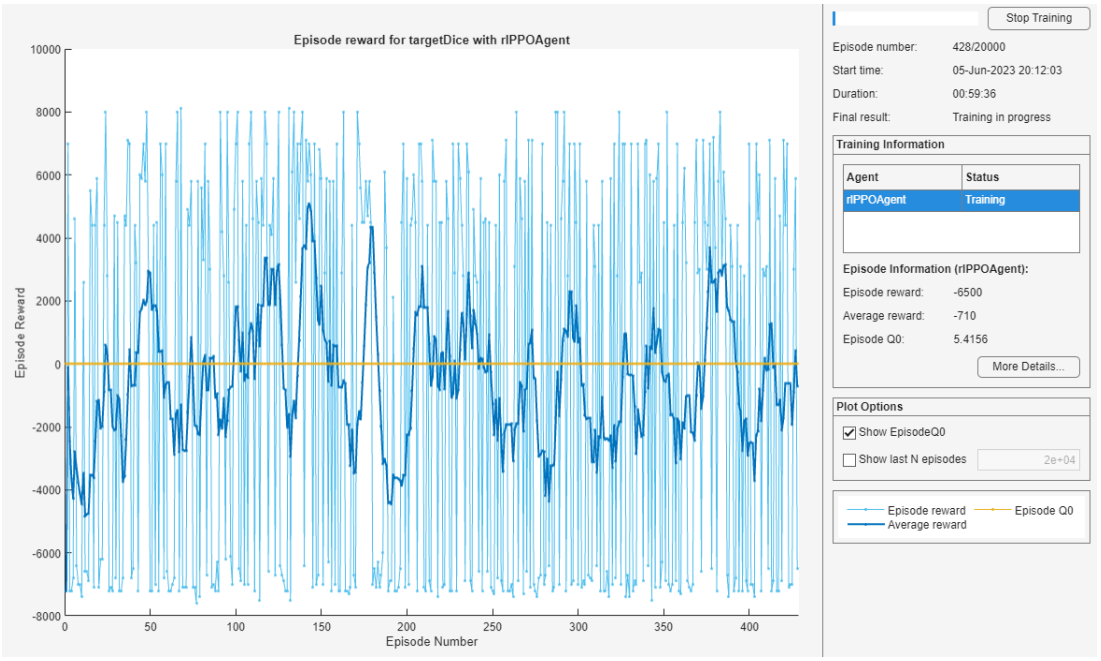
Agentearen aukerak ere modifikatu behar izango balio lehenetsiak egokiak ez direlako [23].

- *“Sample Time”*: Interakzio bakoitzarekiko arteko denbota ezartzen du. Guk 0.2 segundu ezarriko dugu.
- *“Experience Horizon”*: Zenbat urrats egiten beharko ditu agenteak ingurunearekin elkarreraginean ikasketa hasi baino lehen. Gure agenteak 200 experientzia beharko ditu ikasketa hasteko.
- *“Mini Batch Size”*: Gordetako experientzia guztietatik interakzio bakoitzean ikasteko zenbat hartzen ditugun adierazten du. Guk 64 experientzietatik ikasiko dugu interakzio bakoitzean.
- *“Actor Optimizer Options”*: Antzezlearen entrenamendu parametroak zehazteko aukera. Guk antzezlearen aukerak ezartzeko sortu dugun aldagaia ezarriko dugu hemen: actorOpts.
- *“Critic Optimizer Options”*: Kritikoaren entrenamendu parametroak zehazteko aukera. Guk kritikoaren aukerak ezartzeko sortu dugun aldagaia ezarriko dugu hemen: criticOpts.

Bukatzeko, entrenamendu aukerak modifikatuko ditugu.

- "*Maximum Steps Per Episode"*: Guk aldagai hau 10 interakzioetan jarriko dugu. Ondorioz, gehienez kapitulu bat 2s (10 * 0.2s) iraungo ditu.
- *"Max Episodes"*: Kapitulu kantitate maximoa zehazten du. Gure kasuan gehienez 10000 kapitulu izango ditugu.
- *"Score Averaging Window Length"*: Gure kasuan bataz bestekoa azkenengo 20 kapituluekin kalkulatuko da.
- *"Stop Training Criteria , Average Reward, Stop Training Value"*: Entrenamendua gelditzeko bataz besteko saria 10000 izan beharko da.
- *"Save Agent Criteria , Average Reward, Save Agent Value"*: Agente bat gordeko da bataz besteko saria 1300 izaterakoan.

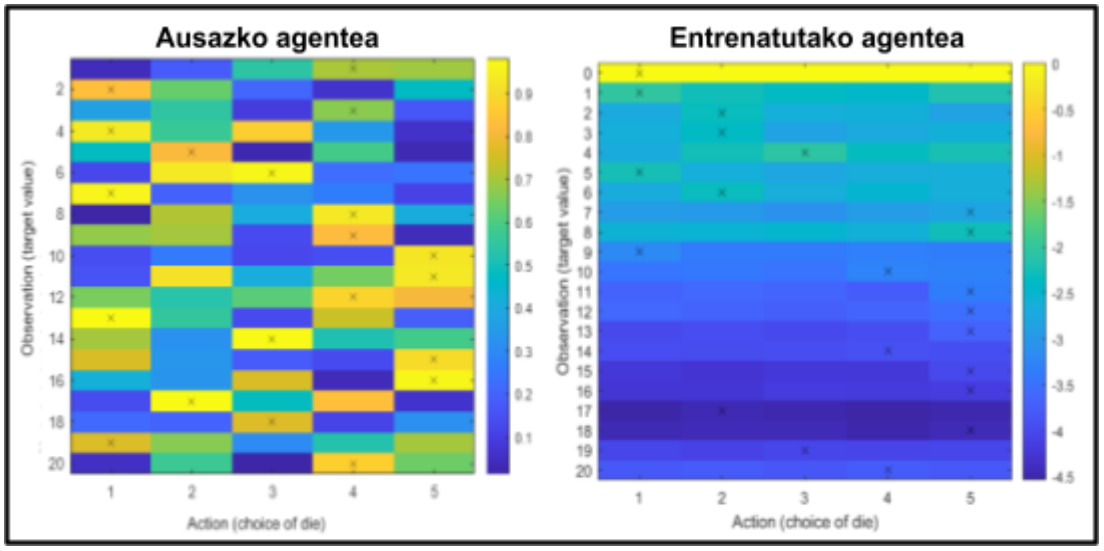
Entrenamendua DDPG agentearekin ikusi genuen egoeraren desberdina izan da (**21.irudia**). Jokoa ausazkoa denez, oso zaila da agenteak ikasketan gorabeherak ez izatea. Ondorioz, **28.irudian** ikusten de moduan gure bataz besteko saria gora beherak izan ditu. Guk momentu altuenetan entrenatutako agenteak gorde ditugu.



28.Irudia: PPO agentearen entrenamendua.

4.1.6.- Erantzuna eta Konprobaketa.

Entrenatutako agentea, ausazko aukeraketak egiten dituen batekin konparatu genuen, gurea hobeagoa zela ziurtatzeko. **29.irudian** ikus daitekeen moduan entrenatutakoa aukeratzen dituen daduak egoera guztietan auzaskoa egiten dituenak baino zentzudunagoak dira.



29.Irudia: Ausazko agente eta entrenatutako agenteare arteko konparaketa.

Hau honela izanda, segurtasunarekin esan dezakegu PPO agente bat sortu eta entrenatu dugula, gure helburua lortzeko.

5. SOFT ACTOR CRITIC (SAC)

SAC algoritmoa, 2018-n Berkeley Unibertsitatearen ikertsaile batzuk propostu zuten [24]. RL munduan egon den azkenengo ideia arrakastatsua izan da.

Metodo hau DDPG algortimoaren antzekoa da. Algoritmo honek duen modifikazio bakarra **entropiaren erregulazioaren kontzeptuan** oinarritzen da.

Reward-ari politikaren entropiari proportzionala den beste sari bat geihtzen zaio ejekuzio denbora bakoitzean. Hau algortimo barruan izanda, politika optimo bat *reward*-a eta entropia balio hau maximizatzen duena izango da. Era honetan, agentearen explorazioa maximizatzen da; Izan ere, agentea extra bat jasotzen du entropia handiko bideak bilatzerakoan.

Horretaz gain, SAC algorimoan askotan beste sare neuronal bat gehitzen zaio. Honek Q-balio funtzioa kalkulatzen du, hau da, kritiko moduan lan egingo du.

Antzezlearen sare neuronala DDPG-n ematen zen eguneraketa bera jasaten du.

6. TWIN DELAYED DEEP DETERMINISTIC

"Twin Delayed Deep Deterministic" edo TD3 2018 aurkeztu zen [25], DDPG algoritmoaren eboluzio baten moduan. TD3-n antzezlearen sare neruonalaren eguneraketa atzeraten da, baina kirtikoarena ez. Era honetan, egileak esaten duten moduan, perturbazioekiko babes handiagoa lortzen da.

30.Irudia: Gaitz txikiak kalte gutxiago eragingo du beti politika bat eguneratzean.

Ondorioz, algoritmoak daraman prosezua honakoa da:

1. Antzezleak lehenengo egoera jasotzen du, s_t.
2. Egoera ikusita antzezleak ekintza bat aukeratzen du, a_t.
3. Ekintzaren saria (reward-a) jasotzen du bueltan, s_{t+1}.
4. Aldi berean, ingurua hurrengo egoerara doa, s_{t+1}.
5. "t" denboran erabilitako politika bera erabilitz hurrengo ekintza aukeratzen da, a_{t+1}.

Horretaz gain, kritikoa bi sare neuronal desberdinak ditu, biak Q-balioaren hurbilketa egiten dutenak. Hartzen den balioa txikiena da, baina gutxiespen bat ez da arazo bat izango; Azken batean, balio hauek algoritmo honetan soilik gradientearen eguneraketan erabiltzen dira. Honek gainestimazioak sortzen dituzten kaltetaz babesten digu; Izan ere, **30.irudian** ikus daitekeen moduan gainestimazio bat, infraestimazio bat baino kaltegarriagoa da.

7. ONDORIOAK

"Reinforcement Learning" tekniken azalpena eta analisia egin dugu euskaraz. Guzti honen oinarria [1] liburua izan da.

Horretaz gain, Matlab softwear-a erabiliz bi agenteen entrenamendu arrakastatsua egitea lortu dugu (ikus **23 eta 29.irudiak**). Matlab-en softwear-a teknika hauen erabilpen praktikoa modu simple batean ezartzeako ahalmena eman digu. Honela, kalkulu asko egin beharrean solik agente, inguru eta sarietan oinarritutako sistema bat eraikiz, agentea atazari modu arrakastatsu batean irtenbidea ateratzea lortu dugu.

Gainera, ataza aldakorra zenean agentea honi adaptatu da, lehenengo adibidean ikusi dugun moduan, non, hasierako posizioa ausazkoa zen.

MatLab-en emandako aukera guztien azalpen simplifikatua ere eman da, honen ulermena errazagoa egiteko.

8. AURREKONTUA:

KONTZEPTUA	Koste Basikoa	Amortizatutako Kostea
Ekipo Informatikoak:		
Ordenagailu pertsonala:*	1.500€	77€
Intel(R) Core(TM) i5-4460 CPU@ 3.20GHz		
8,0 GB DDR3		
NVIDIA GeForce GTX 960		
Acer Aspire 5 A515-51G-59ST*	500€	26€
Programa informatikoak:		
Matlab R2023b	6.000€	309€
Eskulana:		
450 ordu, 50€/h	22.500€	22.500€
GASTU TOTALAK		**22.912€**
Enpresa-gastuak:		
%15 zeharkako kostuak		26.349€
%20 mozkin industriala		31.619€
GUZTIRA, BEZik GABE		**31.619€**
%21 BEZa		38.258€
AURREKONTUA GUZTIRA		**38.258€**
*4 urterako amortizazioa		

Bibliografia

1. Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. TheMIT press, 2014-2015. [Esteka]
2. Maneesh Bhand, Ritvik Mudur, Bipin Suresh, Andrew Saxe, and Andrew Ng. Unsupervised learning models of primary cortical receptive fields and receptive field plasticity. J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 24. Curran Associates, Inc., 2011.
3. Ivan.P.Pavlov. Los reflejos condicionados. 1997 [Esteka]
4. David Silver. Introduction to reinforcement learning with david silver, May 2021. [Esteka]
5. David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. 2014. [Esteka]
6. Bektore espazio. Wikipedia, Entziklopedia askea, 5 urtarrila 2023 . Eskuragarri hemen: https://eu.wikipedia.org/wiki/Bektore_espazio
7. M. Minsky and S. A. Papert. Perceptrons: An Introduction to Computational Geometry. 2017 [Online]. [Esteka].
8. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013. [Esteka]
9. Jordi Torres. Introducción al aprendizaje por refuerzo. Watch This Space, 2021.
10. Percy Huamán Palomino. Programación Dinámica. 2014. [Esteka]
11. Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019 [Esteka]
12. Cornel Secara and Luige Vladareanu. Iterative strategies for obstacle avoidance of a redundant manipulator. Proceedings of the Romanian

Academy - Series A: Mathematics, Physics, Technical Sciences, Information Science, 9, 01 2010.

13. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017
14. rlNumericSpec (MatLab Documentation). Introduced in R2019a. Create continuous action or observation data specifications for reinforcement learning environments. [Esteka]
15. rlSimulinkEnv (MatLab Documentation). Introduced in R2019a. Create a reinforcement learning environment using a dynamic model implemented in Simulink. [Esteka]
16. setVariable (MatLab Documentation). Introduced in R2017a. Set variable value on SimulationInput object.[Esteka]
17. Layer (MatLab Documentation). Introduced in R2016a. Deep learning geruza. [Esteka]
18. rlRepresentationOptions (MatLab Documentation). Introduced in R2019a. Options set for reinforcement learning agent representations (critics and actors). [Esteka]
19. rlDDPGAgentOptions (MatLab Documentation). Introduced in R2019a. Options for DDPG agent. [Esteka]
20. rlTrainingOptions (MatLab Documentation). Introduced in R2019a. Options for training reinforcement learning agents. [Esteka]
21. rlFiniteSetSpec (MatLab Documentation). Introduced in R2019a. Create discrete action or observation data specifications for reinforcement learning environments. [Esteka]
22. rlOptimizerOptions (MatLab Documentation). Introduced in R2022a. Optimization options for actors and critics.[Esteka]
23. rlPPOAgentOptions (MatLab Documentation). Introduced in R2019b. Options for PPO agents. [Esteka]

24. Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018 [Esteka]

25. Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods, 2018 [Esketa]

A ERANSKINA

1. KODEA:

a. DDPG agentearen sorketa eta entrenamendu programa.

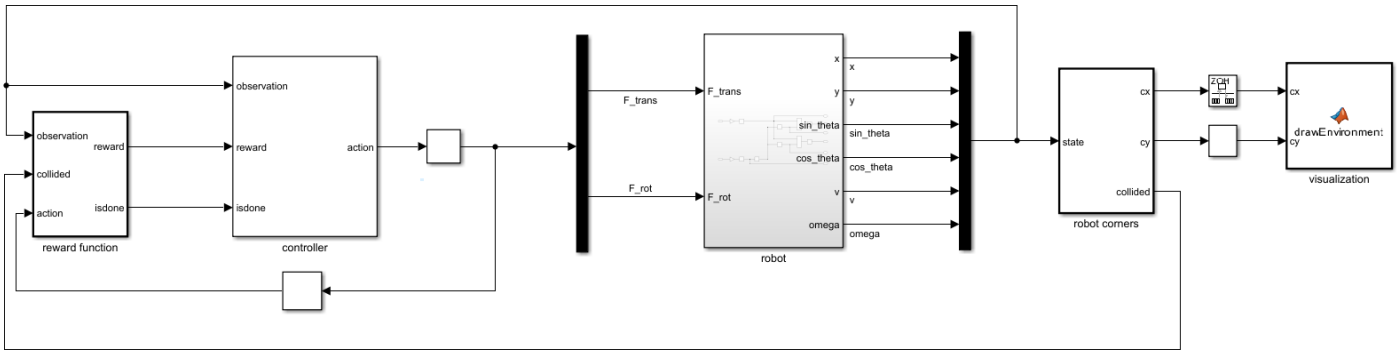
```
Ingurua sortu:
obsInfo = rlNumericSpec([6 1]);
actInfo = rlNumericSpec([2 1],"UpperLimit",1,"LowerLimit",-1);

env=rlSimulinkEnv("DDPGSimulinkEredua","DDPGSimulinkEredua/controller",ob
sInfo,actInfo);
env.ResetFcn = @randomstart;
Sare neuronalak sortu antzezle determinista batententzako:
actnet = [featureInputLayer(6,"Name","obs")
   fullyConnectedLayer(100,"Name","fc1")
   reluLayer("Name","relu1")
   fullyConnectedLayer(100,"Name","fc2")
   reluLayer("Name","relu2")
   fullyConnectedLayer(100,"Name","fc3")
   reluLayer("Name","relu3")
   fullyConnectedLayer(2,"Name","act")
   tanhLayer("Name","scact")];
Aukerak ezarri eta antzezlea sortu:
opts = rlRepresentationOptions("LearnRate",5e-5,"GradientThreshold",10);
actor=rlDeterministicActorRepresentation(actnet,obsInfo,actInfo,"Observation","obs
","Action","scact",opts);
Kritikoa sortu.
Sare neutonala sortu:
obsPath = [featureInputLayer(6,"Name","obs")
   fullyConnectedLayer(100,"Name","fc1")
   reluLayer("Name","relu1")
   fullyConnectedLayer(100,"Name","fc2")
   additionLayer(2,"Name","add")
```

```
    reluLayer("Name","relu2")
    fullyConnectedLayer(100,"Name","fc3")
    reluLayer("Name","relu3")
    fullyConnectedLayer(1,"Name","value")];
actPath = [featureInputLayer(2,"Name","act")
    fullyConnectedLayer(100,"Name","fcact")];
Bi ilarak batu.
valnet = layerGraph(obsPath);
valnet = addLayers(valnet,actPath);
valnet = connectLayers(valnet,"fcact","add/in2");
Kritikoa sortu (antzezlearen aukera berdinekin).
critic=rlQValueRepresentation(valnet,obsInfo,actInfo,"Observation","obs","Action",
"act",opts);
Agentea sortu:
dt = 0.25;
opts=rlDDPGAgentOptions("SampleTime",dt,"ExperienceBufferLength",1e6,"Mini
BatchSize",128);
agent = rlDDPGAgent(actor,critic,opts);

maxsteps = round(30/dt);  % max episode time of 30s (simulation time)
%clear plotRobot          % make sure visualization starts fresh


opts = rlTrainingOptions( ...
        "MaxStepsPerEpisode",maxsteps, ...
        "MaxEpisodes",50000, ...
        "ScoreAveragingWindowLength",50, ...
        "StopTrainingCriteria","AverageReward", ...
        "StopTrainingValue",2);
info = train(agent,env,opts);
```

b. Ausazko hasierako beharrezko kodea.

```
function in = randomstart(in)
mdl = "DDPGSimulinkEredua";
in = setVariable(in,"x0",((-1)^randi([0 1]))*(2.5 + 3.5*rand),"Workspace",mdl);
in = setVariable(in,"y0",2.6 + 3.4*rand,"Workspace",mdl);
in = setVariable(in,"theta0",pi*(2*rand-1),"Workspace",mdl);
in = setVariable(in,"v0",randn/3,"Workspace",mdl);
in = setVariable(in,"w0",randn/3,"Workspace",mdl);
end
```

2. SIMULINK EREDUA.



B ERANSKINA

3. KODEA:

a. PPO agentearen sorketa eta entrenamendua.

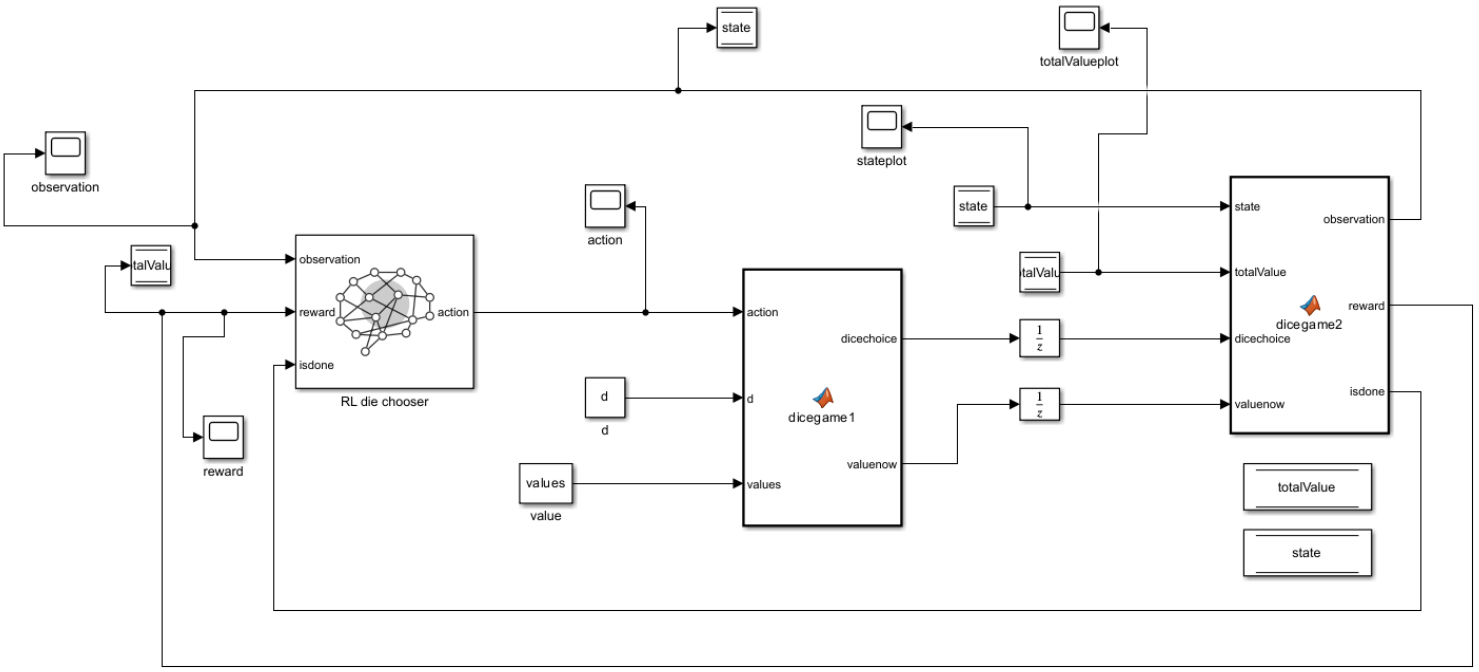
```
clc
clear all
Daduen informazio eta balioak simulinken barnean daude.
Egoeren eta aukeren informazioak zehaztu.
egoerak = 0:20;
aukerak = 1:5;
Daduen informazioa eta balioak.
obsInfo = rlFiniteSetSpec(egoerak);
actInfo = rlFiniteSetSpec(aukerak);
Ingurua sortu.
slEnv = rlSimulinkEnv("PPOSimulinkEredua","PPOSimulinkEredua/RL die
chooser",obsInfo,actInfo);
Antzezlearen sare neuronala sortu.
actnet = [featureInputLayer(1,"Name","target")
  fullyConnectedLayer(10,"Name","hidden")
  reluLayer("Name","relu")
  fullyConnectedLayer(5,"Name","fc")
  softmaxLayer("Name","actprobs")];
Kritikoaren sare neuronala sortu.
valnet = [featureInputLayer(1,"Name","target")
  fullyConnectedLayer(10,"Name","hidden")
  reluLayer("Name","relu")
  fullyConnectedLayer(1,"Name","value")];
Antzezlea sortu.
actorOpts = rlOptimizerOptions(LearnRate=2e-04,GradientThreshold=1);
actor = rlDiscreteCategoricalActor(actnet,obsInfo,actInfo);
Kritikoa sortu.
criticOpts = rlOptimizerOptions(LearnRate=1e-03,GradientThreshold=1);
```

```
critic = rlValueFunction(valnet,obsInfo);
Agentea sortu.
agentOpts = rlPPOAgentOptions(...
    SampleTime=0.2,...
    ExperienceHorizon=200,...
    MiniBatchSize=64,...
    ClipFactor=0.2,...
    EntropyLossWeight=0.01,...
    ActorOptimizerOptions=actorOpts,...
    CriticOptimizerOptions=criticOpts,...
    NumEpoch=3,...
    AdvantageEstimateMethod="gae",...
    GAEFactor=0.95,...
    DiscountFactor=0.998);
agent = rlPPOAgent(actor,critic,agentOpts);
Entrenamenduaren aukerak zehaztu.
diceopts = rlTrainingOptions(...
    MaxEpisodes=10000,...
    MaxStepsPerEpisode=10,...
    Plots="training-progress",...
    StopTrainingCriteria="AverageReward",...
    StopTrainingValue=10000,...
    ScoreAveragingWindowLength=20,...
    SaveAgentCriteria='AverageReward',...
    SaveAgentValue=1300);

info = train(agent,slEnv,diceopts);
```

b. Hasiera bera momentu oro izateko beharrezko kodea.

```
function siminput = randomstart(siminput)
% siminput = setVariable(siminput,"varname",value,"Workspace","modelname");
siminput = setVariable(siminput,"state",20,"Workspace","PPOSimulinkEredua");
siminput = setVariable(siminput,"totalValue",0,"Workspace","PPOSimulinkEredua");
end
```

4. SIMULINK EREDUA



GANTT DIAGRAMA

ATAZA	Hasiera	Bukaera	1.Astea (01/03/2023-05/03/2023)	2.Astea (06/03/2023-12/03/2023)	3.Astea (13/03/2023-19/03/2023)	4.Astea (20/03/2023-26/03/2023)	5.Astea (27/03/2023-02/04/2023)	6.Astea (03/04/2023-09/04/2023)	7.Astea (10/04/2023-16/04/2023)	8.Astea (17/04/2023-23/04/2023)
Informazio eskuraketa eta ikasketa.	1.Astea	3.Astea								
RL-ren ulerketa.	2.Astea	15.Astea								
DDPG agentearen sorketa.	3.Astea	4,Astea								
DDPG agentearen simulink eredua.	4.Astea	6.Astea								
PPO agentearen sorketa.	5.Astea	6.Astea								
PPO agentearen simulink eredua.	6.Astea	7.Astea								
Entrenamenduen frogapena eta optimizazioa	7.Astea	13.Astea								
TFG-aren idazketa osoa.	12.Astea	15.Astea								

ATAZA	Hasiera	Bukaera	9.Astea (24/04/2023-30/04/2023)	10.Astea (01/05/2023-07/05/2023)	11.Astea (08/05/2023-14/05/2023)	12.Astea (15/05/2023-21/05/2023)	13.Astea (22/05/2023-28/05/2023)	14.Astea (29/05/2023-04/06/2023)	15.Astea (05/06/2023-11/06/2023)
Informazio eskuraketa eta ikasketa.	1.Astea	3.Astea							
RL-ren ulerketa.	2.Astea	15.Astea							
DDPG agentearen sorketa.	3.Astea	4,Astea							
DDPG agentearen simulink eredua.	4.Astea	6.Astea							
PPO agentearen sorketa.	5.Astea	6.Astea							
PPO agentearen simulink eredua.	6.Astea	7.Astea							
Entrenamenduen frogapena eta optimizazioa	7.Astea	13.Astea							
TFG-aren idazketa osoa.	12.Astea	15.Astea							