LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

ATDARINOŠĀS MAŠĪNMĀCĪŠANĀS PIELIETOJUMS ROBOTIKĀ

MAĢISTRA KURSA DARBS

Autors: **Pēteris Račinskis**
Stud. apl. Nr. pr20015
Darba vadītājs: Dr. sc. comp. Modris Greitāns

RĪGA 2022

SATURS

1. IEVADS

Sacīt, ka mašīnmācīšanās šobrīd ir ļoti aktuāla pētniecības nozare, būtu maigi. Pēdējās desmitgades laikā tieši šis izpētes lauks ir eksplodējis popularitātē kā neviens cits, pateicoties galvenokārt diviem faktoriem: ļoti vispārīgiem neironu tīklu modeļiem un skaitļošanas resursu veiktspējai, kas beidzot ļāvusi šos teorētiski jau ļoti sen[1, 2, 3] iedomātos mākslīgā intelekta uzbūves elementus realizēt praksē. Tā risināti uzdevumi, ko izsenis daudzi uzskatījuši par neiespējamiem, un lietojuši kā argumentu pret mašīnmācīšanos kā rīku, kas spētu konkurēt ar bioloģiskas izcelsmes prātiem — semantiskas nozīmes meklēšana attēlos[4], tekstu korpusu analīze un ģenerēšana ar "izpratni" par to saturu[5] un visspējīgāko spēlētāju pārspēšana nepilnīgas informācijas spēlēs ar neaptverami milzīgiem iespējamo stāvokļu permutāciju skaitem[6].

Nav arī īpaši grūti atrast vēsturisko saikni starp mākslīgo intelektu un robotiku. Tautas iztēlē termins "robots" drīzāk droši vien iezīmēs zinātniskās fantastikas radītos personāžus — mehāniskas būtnes, kas spēj patstāvīgi darboties neierobežotā vidē un risināt sarežģītus uzdevumus — nevis pieticīgākus, reāli pastāvošus un ražotnēs rodamus industriālos robotus. Un šī pati zinātniskā fantastika radījusi arī nesaraujamu saiti starp robotiem un mākslīgo intelektu[7] — diskusijas par mākslīgo intelektu bieži plūstoši pāriet diskusijās par ar šādu intelektu aprīkotiem robotiem, un šo robotu neizbēgami kareivīgajām ambīcijām attiecībā pret cilvēci. Protams, zinātne ne vienmēr seko populārzinātniskās iedomas lidojumam, taču šāda saikne ir visnotaļ pamatota — spēja mācīties no paraugiem vai patstāvīgi un pielāgoties savai apkārtnei ir ārkārtīgi noderīga, jo daudzi uzdevumi, kuru risināšanai varētu pielietot robotus, ir sarežģīti nevis to fizikālajā izpildē, bet tieši vadības uzdevuma formulēšanā un realizācijā.

Atdarinošā mašīnmācīšanās (*imitation learning*) ir viens no paņēmieniem, ar kuriem tiek mēģināts risināt šādas sarežģītas vadības problēmas. Lai gan pamatu pamatos nevar apgalvot, ka tā ir tikai robotikai piemērota metožu saime, lielākā daļa izpētes virzīta tieši šajā virzienā — problēmas tiek formulētas kā fizikālu (vai nosacīti fizikālu — virtuālās vidēs simulētu) procesu kontroles uzdevumi, un risinājumi tiek rasti no pēc iespējas mazāka skaita veiksmīgas darbības piemēru. Mašīnmācīšanās nozarē bioloģiskas analoģijas un iedvesma nav nekāds retums, un savā ziņā šāda mācīšanās atspoguļo vienu no izplatītiem paņēmieniem, kā cilvēki vai sabiedriski dzīvnieki nodod prasmes viens otram - demonstrējot. Nevar nepieminēt, ka izpēte šajā jomā bieži aizņemas pieejas un iespaidojas no rezultātiem, kas gūti ar stimulēto mašīnmācīšanos (*reinforcement learning*) - savā ziņā vispārīgu, pašmācībai un treniņam analoģisku paņēmienu. Arī abu metožu apvienojums ir ideja, kas pavīd visai regulāri — cerībā, ka, atdarinot ekspertus, var ātrāk nonākt pie derīgām stratēģijām, kas var kalpot kā sākumpunkts dziļākai pašmācībai; vai arī izmantot šādu stimulēto metodi, lai precīzāk imitētu treniņa datus.

1.1. Darba mērķis un struktūra

Šis ir maģistra kursa darbs - pirmais konkrētais rezultāts, kas sasniegts maģistra darba izstrādes procesā. Tāpēc ir jārēķinās ar diezgan īpatnēju formātu un saturu - tiek dokumentēta kāda pētnieciska projekta pirmā fāze, kas bieži vien sastāv no dažādu literatūras avotu izpētes un personiskiem treniņiem, vēl pirms iespējams nopietni sākt eksperimentālu darbību vai pat izvirzīts konkrēts mērķis visam projektam.

Arī šis gadījums nav nekāds izņēmums. Sākumā izvēlēta ļoti aptuvena tēma, balstoties uz Elektronikas un datorzinātņu institūta ekspertu ieteikumiem, un pirmajā darba semestrī lielākoties veikta attiecīgās nozares apguve pašmācības ceļā. Šī nodarbe sastāvējusi galvenokārt no divu veidu darbībām — zinātniskās literatūras lasīšanas un tajā aprakstīto teorētisko jēdzienu un praktisko metožu apguves ar vienkāršiem eksperimentiem personiskās izpratnes veicināšanai.

Līdz ar to šīs atskaites galvenais mērķis ir sniegt ieskatu līdz šim maģistra darba gatavošans ietvaros paveiktajā un apgūtajā. Tā sastāv no trim galvenajām daļām:

1) ievada, kurā īsi izklāstīti vispārīgi jēdzieni, kas nepieciešami, lai izprastu zinātnisko literatūru nozarē;
2) pētniecisku rakstu izlases iztirzājuma un salīdzinājuma;
3) neliela apraksta par paša veikto darbību, apgūstot mašīnmācīšanās modeļus un to realizācijai nepieciešamo programnodrošinājumu.

1.2. Terminoloģijas tulkojumi

Viena no īpatnībām, ar ko ir nācies saskarties, strādājot tieši ar mašīnmācīšanās nozari, ir nepārprotamas terminoloģijas trūkums latviešu valodā. Pati zinātnes nozare, lai arī nebūt ne tik jauna kopumā, piedzīvojusi milzīgas izmaiņas un nepieredzētu uzplaukumu pēdējās desmitgades laikā. Protams, datorzinātnes laukā pirmā un galvenā saziņas valoda ir angļu. Attiecīgi novērajami divējādi un saistīti fenomeni - publikācijas un terminoloģija, kas radītas senāk, veidojušas dziļi specifisku nišu, kas nav iedvesmojusi daudz mēģinājumu tulkot to uz citām valodām, savukārt uzplaukuma laikos vēl ir ļoti daudz materiāla, ko vienkārši neviens nav paguvis iztulkot.

Patvaļīgi izvēloties tulkojumu, pastāv risks mulsināt lasītāju un sadrumstalot jau tā nelielo literatūras kopu dažādu atslēgas vārdu izvēles rezultātā. Tāpēc šeit izveidots saraksts ar potenciāli mulsinoši tulkoto terminoloģiju tās oriģinālajā formulējumā angļu valodā, izvēlētajiem tulkojumiem un īsiem pamatojumiem.

1) *policy* — **stratēģija**. Šis termins pamatā tiek lietots, lai apraksttītu kādu funkciju, kas novērojumus attēlo lēmumu telpā. Pirmais ieraksts tieši tāpēc, ka varētu būt strīdīgākais. Angļu valodā pastāv divi termini, *policy* un *politics*, kas parasti latviski tiek tulkoti vienādi — politika — par spīti radikāli atšķirīgām nozīmēm. Termins *strategy* tiek lietots kā sinonīms pirmajam abās valodās, un arī piemērojams tieši šādām lēmumu pieņemšanas funkcijām, piemēram, spēļu teorijā.
2) *reinforcement learning* — **stimulētā mašīnmācīšanās**. Meklējumi tiešsaistē atklāj

[8], ka šis tulkojums jau ir samērā izplatīts, taču varētu būt nezināms lasītājiem, kas ar to sastopas pirmo reizi — pat ja zināms metodes angliskais nosaukums.

3) *imitation learning* — **atdarinošā mašīnmācīšanās**. Paša autora piedāvāts tulkojums, izmantojot iepriekšējo kā piemēru, jo nav izdevies atrast alternatīvas. Latviskais vārds "atdarināt" izvēlēts pār internacionālismu "imitēt", jo to vieglāk izlocīt formā, kas neizklausās lauzīta un neveikla. Taču procesā zūd spēja viegli atrast sākotnējo vārdu svešvalodā, kas ļoti svarīga zinātniskajā vidē, kurā latviski pieejamo resursu ir maz.

1.3. Tehniskās priekšzināšanas, definīcijas

Pētot un veidojot spriedumus par zinātnisko literatūru viens no lielākajiem šķēršļiem lasītājam "no malas" ir katrā nozarē pieņemtais tehnisko priekšzināšanu kopums, ko autori sagaida no auditorijas. Tas, protams, ir loģiski, jo publikācija, kas apraksta jaunākos atklājumus kādā dziļi specifiskā lauciņā, nevar veltīt visu sev atvēlēto drukas apjomu elementāras un vispārzināmas terminoloģijas skaidrojumiem. Tāpat, tālāk atskaitē iztirzājot šos rakstus, noderīgi ir ieviest tiem kopīgus apzīmējumus un definēt visus vienuviet.

1.3.1. Parametriski modeļi, šabloni

Viens no visplašāk izmantotajiem formālismiem datizraces un mašīnmācīšanās laukos ir parametriskais modelis. Pamatā tam ir ideja, ka nezināmu funkciju, kuras rezultātus vēlamies paredzēt, var aproksimēt ar citu funkciju jeb modeli:

$$M(x) \approx f(x) \tag{1.1}$$

Protams, šādu modeļu varētu būt bezgalīgi daudz, un tie visi var atšķirties pēc tā, cik labi spēj paredzēt nezināmās funkcijas vērtības. Tāpēc modeļu meklēšanai parasti izmanto šablonus - funkcijas, kuru argumentā papildus ievades datiem ir brīvi maināmi un kopīgi (tātad "apmācāmi") parametri θ:

$$\text{Meklē } \theta : M(x|\theta) = M_\theta(x) \approx f(x) \tag{1.2}$$

Iegūtā šablona funkcijas un apmācīto parametru kombinācija $\{M, \theta\}$ tad veido konkrētu modeli. Labs šablons ir tāds, kas spēj pielāgoties ļoti daudzām dažādām funkcijām:

$$\forall f \forall x \exists \theta : M_\theta(x) \approx f(x) \tag{1.3}$$

Atkarībā no uzdevuma specifikas, izplatīti modeļi mēdz būt regresori, kas aproksimē (parasti vektoriālas) funkcijas ar skaitliskām vērtībām,

$$f : x \to \mathbb{R}^k \tag{1.4}$$

$$M : x \times \theta \to \mathbb{R}^k \tag{1.5}$$

un klasifikatori, kas paredz ievades datu punkta piederību kādai diskrētai klasei

$$f : x \to C = \{c_1, c_2, ..., c_m\} \tag{1.6}$$

$$M : x \times \theta \to C \tag{1.7}$$

Bieži vien noderīgi ir ne tikai spēt attēlot datu punktu kā diskrētu klasi, bet iegūt varbūtību sadalījumu, kas apraksta tā iespējamību piederēt jebkurai no klasēm:

$$M : x \times \theta \times c_i \to [0; 1] \tag{1.8}$$

$$M_\theta(x, c_i) = P_i \tag{1.9}$$

$$\sum_{i=1}^{m} P_i = 1 \tag{1.10}$$

Lai varētu novērtēt, cik labi modelis aproksimē nezināmo funkciju, un vadīt parametru apmācības procesu, tiek izmantotas mērķa funkcijas (*loss functions*)[9]:

$$\ell : M_\theta(x) \times f(x) \to \mathbb{R} \tag{1.11}$$

Strādājot ar reāliem datiem, datu punkti veido datu kopu, kas parasti tiek uzskatīta par gadījuma izlasi no punktus ģenerējošā varbūtību sadalījuma. Praktiskiem apmācības uzdevumiem datu kopa parasti jāiegūst formā, kas satur gan sagaidāmos ievades datus, gan pareizu rezultātu:

$$s \sim \mathcal{D} \Leftrightarrow s \text{ ir no varbūtību sadalījuma } \mathcal{D} \tag{1.12}$$

$$y_i = f(x_i) \tag{1.13}$$

$$s_i = (x_i, y_i) \tag{1.14}$$

$$S = \{s_1, s_2, ..., s_n | s_i \sim \mathcal{D}\} \tag{1.15}$$

Datu kopai var aprēķināt empīrisku mērķa funkcijas novērtējumu,

$$L_S(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(M_\theta(x_i), y_i) \tag{1.16}$$

bet apmācības process parasti kādā veidā tiecas minimizēt šīs vērtības matemātisko cerību ģenerējošam sadalījumam (nevis tikai pašai datu kopai - ja modelis ļoti cieši pielāgots konkrētai datu izlasei bet zaudē precizitāti sadalījumam kopumā, to sauc par pārpielāgošanos — *overfitting*)

$$L_\mathcal{D}(\theta) = \mathbb{E}_\mathcal{D}[\ell(M_\theta(x_i), y_i)] \tag{1.17}$$

$$\text{Apmāca } M_\theta \text{ uz } \mathcal{D} \to \text{Minimizē } L_\mathcal{D}(\theta) \tag{1.18}$$

Ja modelis ir stratēģija (*policy*), stimulētās vai atdarinošās mašīnmācīšanās literatūrā to ļoti bieži izsaka kā $\pi_\theta(x)$. Mazliet mulsinošs ir tieši ar imitējošām metodēm saistītos rakstos lietotais apzīmējums π^*, ar ko apzīmē t.s. "ekspertu stratēģijas" — kas pašas ir nezināmās funkcijas, ko cenšamies aproksimēt pēc to ģenerēto punktu kopām.

1.3.2. Neironu tīkli

Neironu tīkls ir izplatīta modeļu šablonu saime, ko var izmantot dažādas formas funkciju aproksimēšanai — tie var būt gan klasifikatori, gan regresori, un pastāv ļoti dažādas to uzbūves variācijas, kas daļēji teorētiski, daļēji empīriskas eksperimentācijas rezultātā un daļēji kopējot bioloģiskās sistēmās atrodamas struktūras izstrādātas dažādu uzdevumu veikšanai. Neironu tīklu kopīgais elements ir t.s. perceptorns, kas izteikts jau pašos pirmsākumos[1]. Perceptrons funkcija, kas piemēro nelineāru aktviācijas funkciju σ argumentu vektora $\vec{x}$ elementu savstarpējai lineārai kombinācijai, t.i,

$$f_{perceptron}(\vec{x}) = \sigma(\vec{w} \cdot \vec{x} + b) \tag{1.19}$$

kur $\vec{w}$ ir t.s. svaru vektors, bet b — nobīde. Perceptrona parametri tātad ir brīvie mainīgie $\vec{w}$ un b. Neironu tīkls parasti sastāv no slāņiem — perceptronu f_i kopām, kas visi apstrādā to pašu argumentu vektoru, bet katrs ar saviem parametriem $\vec{w}_i, b_i$. Tad slāni algebriski izsaka formā

$$W = \begin{bmatrix} w_1^T \\ w_2^T \\ ... \\ w_k^T \end{bmatrix}; \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ ... \\ b_k \end{bmatrix}; \tag{1.20}$$

$$f_{layer}(\vec{x}) = \sigma(W\vec{x} + \vec{b}) \tag{1.21}$$

Ja slānis tīklā ir pēdējais un tā vērtības ir modeļa izvadē, to sauc par izvades (*output*) slāni. Ievades datu vektoru sauc par ievades (*input*) slāni. Pārējos slāņus sauc par slēptajiem (*hidden layers*). Saka, ka slāņi savā starpā pilnīgi savienoti (*fully connected*), ja katram viena slāņa perceptronam argumentā parādās visi iepriekšējā slāņa izvades elementi. Svarīga neironu tīkla īpašība — ja tā aktivācijas funkcijas ir diferencējamas, tad arī tīkls kopumā ir diferencējams pēc katra tā parametra, pat ar perceptroniem daudzos slāņos. Līdz ar to var izmantot t.s. *backpropagation* algoritmu, kas atrod mērķa funkcijas parciālos atvasinājumus pēc modeļa parametriem un izmanto kādu gradientu optimizācijas metodi apmācībai.

Pastāv dažādas šo tīklu arhitektūras. Vienkāršākās sastāv no viena vai vairākiem slāņiem (neskaitot ievades slāni), taču ir plaši izplatīti arī, piemēram, konvolucionālie neironu tīkli[4], ko izmanto attēlu apstrādē, tai skaitā šajā atskaitē aplūkotajos pētījumos, kur nepieciešams gūt informāciju no video datiem. Galvenā atšķirība konvolucionālajā tīklā ir t.s. kodola funkciju jeb kerneļu (*kernel*) izmantošana - konvolucionāli slāņi vienā līmenī piemēro identiskas perceptrona funkcijas nelieliem iepriekšējā slāņa (matricas vai tenzora formā) reģioniem. Tas palīdz identificēt dažādas lokālas struktūras, piemēram, attēlā. Šo un vēl citu veidu sarežģītāku neironu tīklu arhitektūra ir ļoti plašs lauks, ko detalizēti šeit iztirzāt nav iespējams.

1.3.3. Markova lēmumu procesi

Pastāv dažādi formālismi procesu definēšanai vadības sistēmu izstrādes mērķiem, lai ar tiem varētu veikt matemātiskas operācijas. Izplatīti atdarinošās un stimulētās

mašīnmācīšanās literatūrā ir Markova lēmumu procesi (MDP — *Markov decision processes*), kas izmantojami situācijās, kad sistēmas stāvokli nākotnē pilnībā nosaka pašreizējais. Dažādi autori, kas darbojas dažādos izpētes virzienos, mēdz piedāvāt dažādus tā formulējumus, taču parasti tie ir ekvivalenti sekojošam[10]

$$MDP = (S, A, R, T, \gamma) \tag{1.22}$$

kur S — sistēmas iespējamo stāvokļu s kopa; A — kontrolētajam procesam ("aģentam") pieejamo darbību a kopa; $R : S \times A \to \mathbb{R}$ vai $R : S \to \mathbb{R}$— atdeves (*reward*) funkcija, kas ļauj kārtot sasniegtos stāvokļus pēc to tīkamības; $T : S \times A \to S$ vai $P(s' \in S)$ — pārejas (*transition*) funkcija, kas nosaka nākamo stāvokli s' vai tam atbilstošu varbūtību sadalījumu, ja pie iepriekšējā stāvokļa s izvēlēta darbība a; γ — koeficients nākotnes atdevju vērtību samazināšanai. MDP ir *galīgs* ja S, A ir galīgas kopas. Ja $s' = T(s, a)$ ir determinēts, MDP ir *determinēts*. Ja s' ir gadījuma lielums, kas pieder sadalījumam $P(s') = T(s, a)$, MDP ir *stohastisks*.

Atdarinošās mašīnmācīšanās metodēm ne vienmēr ir nepieciešams definēt atdeves funkciju un attiecīgi arī γ, taču tie ir nepieciešami metodēm, kas lieto stimulēto mašīnmācīšanos. Tā kā parasti spriests tiek par stratēģijām π_θ, kas izvēlas nākamo darbību a atkarībā no sistēmas stāvokļa s, tad bieži vien faktiskā pārejas funkcija ir formā $P(s') = T(s, \pi_\theta(s), s')$, t.i., pārejas funkcija apraksta "vides" (*environment*) reakciju uz aģenta (modeļa, stratēģijas) darbību. Pie sākotnējo stāvokļu kopas S_0 un stratēģijas π var spriest par stratēģijas inducēto stāvokļu sadalījumu $s_t \sim P(S|S_0, \pi)$ — tas nosaka, kādas trajektorijas vispār ir iespējamas pie šādiem nosacījumiem. Ļoti izplatītas ir arī situācijas, kad modelis ņem vērā nevis pilno sistēmas stāvokli, bet gan t.s. novērojumu (*observation*) — $\pi_\theta(o) = \pi_\theta(g(s))$. Tā ir funkcija no kādas stāvokli raksturojošo parametru apakškopas, un bieži vien ļoti nepilnīgi šo stāvokli raksturo.

Trajektoriju, kādai process seko ar laika soļiem $t = \{1, 2, ..., T\}$, raksturo laikrinda (*state-action*) pāru formā — $((s_1, a_1), (s_2, a_2), ..., (s_T, a_T))$. Stāvokļus tajā, protams, iespējams aizstāt ar novērojumiem situācijās, kad tiek izmantota nepilnīga informācija. Ne vienmēr vēlams vai iespējams modelēt sistēmu ar MDP. Ir iespējami gadījumi, kad pārejas funkcija vai stratēģija ir atkarīga no laika soļa, kā arī sistēmas, kurās ar novērojumiem nepietiek lēmuma pieņemšanai un nepieciešams ņemt vērā iepriekšējo stāvokļu un darbību virkni, lai pareizi spriestu par slēptiem stāvokļa atribūtiem.

1.3.4. Stimulētā mašīnmācīšanās

Stimulētā mašīnmācīšanās ir pati par sevi ļoti aktuāla izpētes nozare, un nereti nodarbojas ar to pašu vai līdzīgu uzdevumu risināšanu, kā atdarinošā. Pastāv ne tikai kombinēti paņēmieni[11, 12], bet arī atdarināšanas metodes, kas tiešā veidā izmanto stimulēto mācīšanos, lai atdarinātu trajektoriju demonstrācijas[13]. Tāpēc nav nekāds pārsteigums, ka šis termins visnotaļ bieži parādās ar atdarinošo mašīnmācīšanos saistītos pētījumos, citreiz bez nekādiem papildus paskaidrojumiem.

Stimulētās mašīnmācīšanās teorētiskie pamati ir galīgi MDP un Belmana vienādojums[14]. Pieņem, ka katram stāvoklim ir kāda atdeve $R(s_t)$, bet uzdevums — maksimizēt

šo atdevju summu visā trajektorijas garumā $\sum_{t=1}^{T} R(s_t)$. Tad var izteikt arī varbūtību sadalījumu atdevei katram stāvokļa un darbības pārim

$$p(s', r|s_t, a_t) = P[s_{t+1} = s', r = R(s')] \tag{1.23}$$

Nākotnē sagaidāmās atdeves vērtības, ņemot vērā dilšanas koeficientu γ, var izteikt kā

$$G_t = \sum_{k=0}^{T-t} \gamma^k R(s_{t+k+1}) \tag{1.24}$$

Jebkura stratēģija katram stāvoklim nosaka darbību vai darbību sadalījumu $p(a|s) = \pi(a, s)$. Var izmantot rekursīvu sakarību, lai katram stāvoklim piekārtotu sagaidāmo atdevi jeb vērtību $v_\pi(s)$, kas atkarīga no izmantotās stratēģijas — Belmana vienādojumu.

$$v_\pi(s) = \mathbb{E}_\pi[G_t|s_t = s] = \sum_a \pi(a, s) \sum_{s',r} p(s', r|s_t, a_t)[r + \gamma v_\pi(s')] \tag{1.25}$$

Atrisināt mācīšanās uzdevumu tādā gadījumā nozīmē atrast stratēģiju, kas maksimizē atdevi. Pastāv dažādas metodes, kā to darīt. Teorētiski vienkāršākais taču praktiskiem uzdevumiem reti piemērojams paņēmiens ir tā saucamā Q-mācīšanās. Tā strādā samērā vienkārši — tiek izveidots tenzors Q ar elementu, kas atbilst katrai iespējamai (s, a) vērtībai, tam tiek piešķirta kāda sākotnējā vērtība (piemēram, 0).

Apmācība notiek, izvēloties

$$a_t = \max_a(Q(s_t = s)) \tag{1.26}$$

un sasniedzot trajektorijas beigas — vai nu pēc noteikta soļu skaita T, vai arī kāda pārtraukšanas nosacījuma. Tad iegūtajai trajektorijai $(s_1, a_1), (s_2, a_2), ..., (s_T, a_T)$ no beigām aprēķinot $G_1, G_2, ...G_T$ atbilstoši katram solim var koriģēt vērtības tenzorā

$$Q^{i+1}(s_t, a_t) = f(Q^i(s_t, a_t), G_t) \tag{1.27}$$

Kaut gan šai metodei ir teorētiskas konverģences garantijas pēc pietiekama iterāciju skaita, ļoti strauji pieaug tās modeļa — tenzora Q — parametru skaits, pieaugot iespējamo stāvokļu un darbību skaitam — nepieciešams atsevišķi optimizēt katru iespējamo kombināciju, iespējams, ļoti daudzās iterācijās. Tāpēc praksē parasti tiek lietoti modeļi, kas aproksimē $v_\pi(s)$, piemēram, aģenta-kritiķa (*actor-critic*) neironu tīkli, kas reizē iemācās paredzēt gan sagaidāmo vērtību, gan labāko darbību katram stāvoklim ar potenciāli daudz kompaktāku modeli.

Lai uzdevumu varētu risināt, nepieciešams spēt izteikt kādu analītisku funkciju, kas apraksta pašreizējā stāvokļa tīkamību — izšķir labus rezultātus no sliktiem, vai starpstāvokļiem. Robotikā var būt sarežģīti šādu funkciju izdomāt, turklāt tā var būt ļoti "retināta" stāvokļu-darbību telpā, t.i., tikai ļoti nelielam skaitam (vai ar ļoti nelielu varbūtību) sasniegto stāvokļu atdeves funkcija $R(s_t)$ pieņem nenulles vērtību. Tieši šādu

trūkumu mēģina risināt metodes, kas kombinē ekspertu demonstrāciju aproksi-mēšanu ar adaptīvu pielāgošanos[15]

1.3.5. Robotikas uzdevumi

Darbā ar robotiem uzdevums parasti ir vēlamas paša robota un citu vidē atrodamo objektu telpiskās konfigurācijas sasniegšana, vai virkne ar šādām pārejām (manipulācijām). Protams, pilnīgu fizikālas vides pašreizējā stāvokļa aprakstu gūt nav iespējams, tāpēc trajektoriju laikrindas vienmēr īstenībā sastāvēs no novērojumiem, nevis stāvokļiem — $((o_1, a_1), (o_2, a_2), ...)$. Novērojumu formas var būt ļoti dažādas — sākot ar ļoti detalizētiem robota izpildelementu konfigurācijas (lineāro vai lenķisko pārvietojumu, ātrumu, paātrinājumu, slodžu) aprakstiem, beidzot ar video bez nekādas anotācijas.

1.4. Pētniecības virzienu tematisks dalījums

Varētu sacīt, ka tieši par atdarinošo mašīnmāīšanos rakstīts ir samērā maz. Noteikti, ja salīdzina ar vispārīgākām metodēm vai rīkiem. Taču pat "samērā maz" tomēr nozīmē ļoti lielu publikāciju skaitu, kas apraksta pētījumus ļoti dažādos virzienos. Turklāt robotika dominē kā pielietojuma mērķis šādām metodēm. Lai radītu priekštatu par nozares pašrei-zējo stāvokli un aptuvenu vēsturi, nolemts izšķirt trīs aptuvenus virzienus, kas labi apraksta lielu daļu no pētījumiem par iespējām robotus apmācīt ar piemēriem:

1) trajektoriju kopēšana — mērķi šeit pamatā ir panākt robustu, precīzu atdarināšanu ar nelielām treniņa datu kopām, ja pieejama nepieciešamā informācija par sistēmas stāvokli;
2) novērojumu atdarināšana — ne vienmēr ir pieejami dati padevīgā formā, lai tiešā veidā varētu imitēt tajos veiktās darbības. Plaša pētījumu joma nodarbojas tieši ar trajektoriju iegūšanu no video datiem;
3) adaptīvu un atdarinošu metožu kombinācija — atdarinošās mācīšanās pielietojums, lai uzlabotu stimulēto, un otrādi. Kā panākt, ka neaprobežojamies ar tikai piemēros esošo un spējam pielāgoties? Kā efektīvi uzsākt stimulēto mācīšanos ļoti retinātās atdevju telpās?

2. LĪDZŠINĒJIE PĒTĪJUMI

Šīs nodaļas mērķis ir izveidot aptuvenu nozares pētniecības vēsturisku pārskatu; aprakstīt galvenos sasniegtos rezultātus, gūtās atziņas katrā no tematiskajiem apakšvirzieniem. Protams, ne visus pētījumus iespējams vienkārši klasificēt pēc to piederības šeit izvēlētajām kategorijām, un daudzi varbūt tajā vispār neiederas — taču cenšoties gūt personisku izpratni par kādu tēmu, lai motivētu tālākus pētījumus, ir svarīgi nostatīt iepriekšējus rezultātus to kontekstā, saprast, kāpēc tieši šobrīd aktuālie pētniecības virzieni ir tādi, kādus tos varam redzēt kādā akadēmisko publikāciju datubāzē vai neseno pētījumu pārskatā.

Savā ziņā varētu teikt, ka trīs nodaļās nostādītie mērķi kopā būvē pamatus atdarināšanai kā praktiski izmantojamai modeļu apmācības metodei — vai vismaz tādu priekštatu ir ērti sev radīt, lai labāk orientētos savstarpējās atkarības attiecībās starp to sasniegšanai veltīto pētījumu rezultātiem.

2.1. Trajektoriju kopēšana

Pirmā, varētu teikt galvenā taču ne vienmēr vienkāršākā problēma, ir atrast veidu, kā piejamās ekspertu zināšanas — robotikas kontekstā tās parasti būs pareizas trajektorijas dažādu pārvietojumu un smalku manpiulācijas uzdevumu risināšanai — tiešā veidā atdarināt. Šo procesu mēdz saukt arī par programmēšanu ar demonstrācijām (PBD — *programming by demonstration*)[16, 17]. Idealizētā vidē ar determinētām stāvokļu pārejām un pilnīgu informāciju par tās pašreizējo konfigurāciju šis uzdevums varētu būt pat triviāls, taču praksē saskaramies ar problēmām:

1) darbs notiek ar novērojumiem, nevis stāvokļiem. Pat ja pieejami, piemēram, trajektoriju ieraksti, bieži vien trūkst svarīgas informācijas (varētu būt zināma trajektorijas kinemātika, bet ne tās dinamika — paātrinājumi, bet ne spēki);
2) atšķirības vidē: izpildelementos — varbūt robots ir nedaudz citāds; apkārtnē — varbūt manipulējamo objektu masas, forma vai izvietojums ir nedaudz atšķirīgi no demonstrācijās esošajiem;
3) ja trajektoriju ģenerējis eksperts, kam, iespējams bijusi pieejama informācija, kuras aģentam nav — piemēram, manipulāciju veicis cilvēks ar redzi, bet robotam pieejami tikai kontakta sensori.

Problēmas faktiski nozīmē to, ka reālā sistēmā stāvokļu pārejas nav determinētas attiecībā pret novērojumiem un darbībām. Lai labāk saprastu šos trūkumus, vispirms noderīgi ir aplūkot "naivākos" veidus, kā varētu imitēt piemērus.

2.1.1. Vienkāršas metodes

Pirmais, ko varētu darīt, ir tiešā veidā ierakstīt trajektoriju un to atkārtot. Šī nebūt nav jauna ideja — gandrīz visiem mūsdienu industriāliem robotiem ir pieejamas t.s. *lead-through* un *teach-in* programmēšanas metodes, kas ļauj fiziski un ar tālvadības ierīces palīdzību vadīt robota kustību un to ierakstīt pēcākai atdarināšanai[18], turklāt tās parādījušās jau pašos industriālās robotikas pirmsākumos 1970os gados[19].

Darba autors var pats personīgi izdarīt zināmus secinājumus par tiešu trajektoriju ierakstīšanu un atkārtošanu, jo ir strādājis kā mehatronikas inženieris uzņēmumā, kas nodarbojas ar rūpnieciskās ražošanas iekārtu projektēšanu, izgatavošanu un automatizāciju, tāpēc pietiekami daudz nodarbojies arī ar robotu programmēšanu. Tā kā trajektorijai jābūt ierakstītai tieši ar robotu, lai tā būtu atkārtojama bez papildus datizraces uzdevumu risināšanas, ir zināma tendence dominēt viegli realizējamiem bet varbūt ne optimāliem ceļiem telpā — vieglāk ierakstīt dažus pagrieziena punktus un ļaut programmatūrai interpolēt nekā fiziski vadīt robotu visā kustības ceļā.

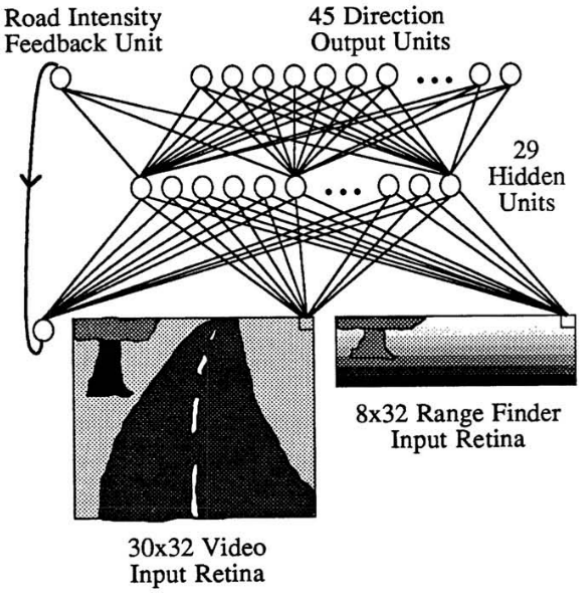
Turklāt var parādīties neparedzēti trūkumi, pārejot no lēnas, nenoslogotas izpildes programmēšanas procesā uz ātru un noslogotu ekspluatācijā, kas apgrūtina procesu. Faktiski sākotnējais ieraksts bieži vien kalpo par starta punktu, bet, lai nonāktu pie lietojamas programmas, nepieciešams iegūto kodu koriģēt un iteratīvi pielāgot. Lai arī principā tiek izmantota demonstrācija trajektorijas iegūšanai, procesa veikšanai tik un tā nepieciešams personāls ar robotu programmēšanas prasmēm. Jau sen atzīts[16, 17], ka, lai tik tiešām robotus varētu apmācīt tikai ar piemēriem, nepieciešamas metodes, kas ir robustākas pret nobīdēm no paraugu ģenerējošā procesa apstākļiem, vispārināmākas, un attiecīgi sākti pētījumi ar mašīnmācīšanās metodēm.

Kad jāspēj atdarināt kas vairāk nekā viena, nemainīga trajektorija, nepieciešams atdarināt nevis pašu trajektoriju, bet gan procesu, kas tādas ģenerē — "eksperta" stratēģiju. Viena no vienkāršākajām metodēm, kas bieži tiek lietots kā piemērs, taču praksē reti kad ir pielietojuma, ir uzvedības klonēšana (*behavioural cloning*). Vispārīgi to definēt ir samērā vienkārši[10]. Ja dots MDP un kāda eksperta stratēģija π^*, kas šo MDP optimāli risina, mērķis ir atrast maksimāli tuvu modeli π_θ, kur

$$\pi_\theta(s) \approx \pi^*(s) \tag{2.1}$$

Parasti, protams, ir pieejama datu kopa ar eksperta izietajām stāvokļu-darbību laikrindām, turklāt jāstrādā ir ar novērojumiem, nevis stāvokļiem. Kā ilustratīvu piemēru mēģinājumam realizēt šādu algoritmu bez īpašām korekcijām var izmantot 1989. gadā Kārnegija-Melona Universitātē veikto pētījumu "*Autonomous Land Vehicle in a Neural Network*" (ALVINN)[20]. Tā mērķis bija izstrādāt pašbraucošu automašīnu, kas spēj sekot ceļa kontūram.

Automašīna tikusi aprīkota ar videokameru un LIDAR sensoriem, kas devuši divus skatus uz to pašu telpas reģionu automobiļa priekšā. Par apmācāmo modeli izvēlēts neironu tīkls. Protams, 1989. gads vēl bija laiks, kad datoru veikstpēja bija stipri ierobežota, un nevienam vēl nebija ienācis prātā būvēt tik dziļas, daudzskaitlīgas un sarežģītas tīklu arhitektūras kā mūsdienu konvolucionālos tīklus vai transformatorus. Tāpēc neironu tīkls ir gaužām līdzīgs jebkurā mācību grāmatā pirmajā nodaļā atrodamajiem piemēriem — tam ir viens slēptais slānis ar 29 perceptroniem, kam seko 45 izvades elementi. Video izmantots krāsainā attēla zilais kanāls, jo tajā ceļa virsma visvairāk kontrastē ar apkārtējo vidi. Gan video, gan LIDAR radītie attēli tīkla ievadē veido vienkāršu vektoru bez nekādiem telpiskiem kodējumiem, visi slāņi savstarpēji pilnībā savienoti.



Att. 1: ALVINN modeļa uzbūve[20]

Modeļa izvades slānis apzīmē vēlamo stūrēšanas virzienu 45 diskrētos soļos. Treniņa datu kopā faktisko virziena komandu atspoguļo neprecizēta veida "zvana" funkcija ar modu pie pareizā virziena. Ieviests viens papildus perceptrons, kas (teorētiski) novērtē ceļa gaišumu salīdzinot ar apkārtējo vidi, un tiek pievienots nākamās iterācijas ievades vektoram.

Jau šim (šķietami) samērā vienkāršajam uzdevumam konstatēts, ka ievākt treniņa datus fizikālā vidē — braucot ar automašīnu pa ceļiem un ierakstot vadītāja veiktās korekcijas — nav praktiski, jo nepieciešama ļoti liela treniņa datu kopa. Jāatzīst, ka ar modernākiem tehniskās redzes modeļiem droši vien šī nepieciešamība mazinātos. Tāpēc dati ģenerēti sintētiski — tā kā gan video, gan attāluma datu izšķirtspēja ir gaužām neliela, pat ar 1989. gadā pieejamām datorgrafikas iespējām šādi gūtus attēlus ir grūti atšķirt no īstiem. Simulatorā iegūtie attēli un vadības komandas izmantoti klasifikatora apmācībā.

Iegūtais rezultāts — modelis, kas maksimāli tuvināts simulatorā realizētajam kontroles algoritmam izmantotā šablona iespēju robežās. Tas bijis pietiekami labs, lai spētu vadīt ar kameru un attāluma sensoru aprīkotu automobili pa 400m garu slēgta ceļa posmu saulainos dienas apstākļos, ar ātrumu 0,5m/s. Tas tiek lietots kā arguments par neironu tīklu pavērtajām iespējām pašbraucošo auto attīstībā, taču netiek slēpts, ka sasniegtais ir tālu no praktiskas vadības sistēmas.

Kā galvenais uzvedības klonēšanas trūkums parasti tiek minēta nespēja atgūties no faktiskā stāvokļa sadalījumu nobīdes[10] (*distribution shift*). Ja reālais modelis $\pi_\theta(s)$ nevar pilnīgi precīzi atdarināt eksperta $\pi^*(s)$ darbības vai MDP pārejas funkcija ir stohastiska, tātad treniņa datu kopa neietver visas iespējamās trajektorijas ar atbilstošajām $\pi^*(s)$ vērtībām, π_θ inducētais stāvokļu sadalījums diverģē no π^* inducētā. Lai iegūtu precīzāku un robustāku eksperta stratēģijas atdarinājumu, piedāvāti dažādi — sarežģītāki — apmācības paņēmieni.

2.1.2. Statistiskas korekcijas

Viens no virzieniem, kurā vesti centieni uzlabot trajektorju kopēšanas lietderību, ir strukturēt treniņa datu ieguvi un apmācības algoritmu veidā, kas maksimāli tuvina π^* un π_θ inducētos stāvokļu sadalījumus. Bieži vien tas nozīmē, ka vienkārši ievākt datus un veikt apmācību uz tiem vairs nav iespējams — nepieciešama aktīva instruktora iesaiste. Piedāvātie risinājumi ir dažādi, un metodes var kļūt visnotaļ sarežģītas[10], tāpēc, lai ilustrētu pieejas būtību kopumā, izvēlēts viens, vairāk teorētisks piemērs.

DAgger — *dataset aggregation* — ir 2011. gadā publicētajā rakstā "*A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning*"[21], kā autori atkal ir no Kārnegija-Melona Universitātes ASV, piedāvāts algoritms. Tas piedāvā teorētiskas garantijas π^* un π_θ inducēto sadalījumu konverģencei, kombinējot instruktāžu ar apmācāmā modeļa ģenerētām stratēģijām. Lai gan raksts nedarbojas tieši ar robotiku, izmantotais MDP kontroles formālisms ir vispārīgs.

Algoritma darbību var vienkāršoti aprakstīt sekojoši: pieņem, ka ir pieejami ne tikai eksperta ģenerēti trajektoriju dati, bet ir iespējams pašam ekspertam uzdot vaicājumus par katrā stāvoklī optimālu darbību — kas, ja instruktāžu nodrošina cilvēks un laika soļu pārejas ir biežas, reti kad būs praktiski iespējams. Tādā gadījumā iteratīvi atkārto šādus soļus:

1) ar kādu varbūtību α izvēlas, vai i-tā trajektorija tiks ģenerēta ar π^* vai π_θ;
2) iegūtās laikrindas stāvokļa elementiem s_t atrod atbilstošo $a_t^* = \pi^*(s_t)$
3) kopējai datu kopai D pievieno $D_i = \{(s_1, a_1^*), ...\}$
4) apmāca modeli π_θ uz papildinātās datu kopas

Kā jau minēts, liela daļa raksta satura veltīta tieši algoritma teorētisko īpašību pierādīšanai, taču beigās arī veikti daži eksperimenti — divi ar personāžu vadību datorspēļu vidē, viens ar rokraksta zīmju atpazīšanu teksta virknēs. Lai gan visos gadījumos DAgger pārspēj uzvedības klonēšanas (vienkāršas π^* aproksimēšanas no treniņa datu kopas) rezultātus, tā lietderību stipri ierobežo instruktora interaktivitātes prasības — praksē reti kad ir iespējams kaut kas analoģisks datorspēļu aģentu treniņam izmantotajam simulatoram, kas ar dziļu pārlasi atrod labas stratēģijas jebkuram stāvoklim.

2.1.3. Inversā stimulētā mācīšanās (IRL)

Cits veids, kā atdarināt instruktora dotas trajektorijas, ir pieņemt, kas tā stratēģija optimizē kādu slēptu atdeves funkciju $R^*(s)$ un mēģināt to atjaunot no pieejamās informācijas. Šādā veidā ar stimulētās mašīnmācīšanās metodēm var iegūt meklēto rezultātu. Kā jau parasti, iespējami dažādi veidi, kā formalizēt uzdevumu un tehniski to realizēt. 2004. izdotais "*Apprenticeship learning via inverse reinforcement learning*"[22] no Džordžijas Tehnoloģiju institūta, viens no citētākajiem rakstiem par šo tēmu (lai arī ne pirmais), piedāvā iteratīvu algoritmu nezināmas atdeves funkcijas atjaunošanai un izmantošanai. Galvenā atkāpe no tipiska MDP formālisma ir pieņēmums, ka nezināmā atdeves funkcija R^* ir formā,

$$R^*(s) = w^* \cdot \phi(s) \tag{2.2}$$

kur w^* — svaru vektors, bet $\phi : S \to [0,1]^k$ — zināmu atribūtu izpausme noteiktos stāvokļos. ϕ nozīme ir tāda, ka ir iespējams noteikt, kādu sakarību lineāra kombinācija varētu būt īstā atdeves funkcija. Kā piemērs tiek piedāvāts autovadītāja uzdevums — viens no atribūtiem varētu būt 1, ja mašīna atrodas uz ceļa, bet 0 citādi, u.t.t. m treniņa kopas trajektorijām aprēķina vidējo faktoru vērtību summu, izteiktu kā

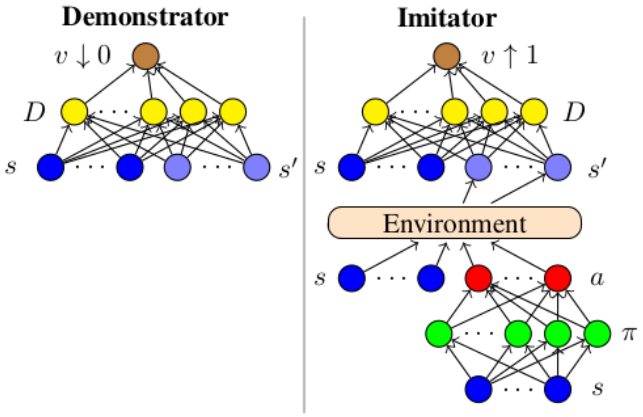
$$\mu^* = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{T} \gamma^t \phi(s_t^i) \tag{2.3}$$

Tad tiek iteratīvi atkārtota procedūra, kur atrod kādu svaru vektoru w^i un attiecīgi empīrisku atdeves funkciju $R^i(s) = {w^i}^T \phi(s)$, ko izmanto, lai apmācītu jaunu stratēģiju π^i. Tad šai stratēģijai atrod vidējo vērtību μ^i analoģiski (2.3), un visas iepriekšējās $\mu^{j \leq i}$ tiek izmantotas, lai atrastu nākamo svaru vektoru w^{i+1}. Process turpinās, līdz ir konverģējis līdz noteiktam kļūdas hiperparametram. Tādējādi beigās iegūta stratēģija, kas maksimizē līdzīgu atribūtu $\phi(s)$ kombināciju nezināmajai instruktora stratēģijai, un robusti seko demonstrācijām.

Rezultāti parāda, ka šī metode pārspēj dažādas vienkāršas, statistiskas π^* aproksimācijas metodes (uzvedības klonēšanu). Kopā risināti divi dažādi uzdevumi. Viens ir "*gridworld*" — spēle, kurā aģents pārvietojas pa režģa formas vidi un dažos lauciņos ir pieejamas atdeves. Taču pārejas process ir stohastisks, tāpēc metodes, kas atdarina tikai telpiskos pārvietojumus un nemēģina atjaunot slēpto atdeves funkciju darbojas sliktāk. Otrs ir divdimensionāla spēle, kurā aģents vada automobili. Šeit tika pārbaudīts, vai var iemācīt aģentam atšķirīgus "braukšanas stilus" tikai ar demonstrācijām, kas arī izdevies.

2.1.4. Ģeneratīvie sāncenšu tīkli

Iedvesmojoties no inversās stimulētās mācīšanās, attīstītas arī citas metodes, kas tiešā vai netiešā veidā nodarbojas ar instruktora stratēģijas aproksimēšanu. Bieži kā trūkums IRL metodēm tiek minēta nepieciešamība katrai iteratīvi iegūtajai stratēģija no jauna veikt stimulēto apmācības procesu, lai būtu iespējams iegūt šīs metodes ģenerētas



Att. 2: GAN uzbūves piemērs. Augšējais tīkls — diskriminators — cenšas atšķirt eksperta demonstrācijas no ģeneratora radītām trajektorijām[23]

trajektorijas un līdz ar to varētu novērtēt to sasniegto stāvokļu atribūtu sadalījumus. Meklējot veidus, kā tiešā veidā optimizēt stratēģiju, lai tā sasniegtu tādus pašus novērtējumus kā demonstrācijas pie plašām iespējamo atdeves funkciju klasēm, izlaižot pašu šo atdeves funkciju meklēšanu, 2016. gadā publicēts "*Generative Adversarial Imitation Learning*"[24]. Tas piedāvā risināt trajektoriju atdarināšanas uzdevumu ar sāncenšu tīklu metodi, un ir bijis samērā ietekmīgs uz tālākiem pētījumiem nozarē, jo šobrīd tā jau ir visai populāra metode, un vēl salīdzinoši nesen tika uzskatīta par labāko tieši trajektoriju kopēšanas uzdevumā[23].

Ģeneratīvie sāncenšu tīkli — GAN, *generative adversarial networks* — ir neironu tīkli, kas apmācīti visai īpatnējā veidā. Tā vietā, lai optimizētu visu modeli vienam mērķim, tiek izdalīti divi elementi — ģenerators un diskriminators — ar pretējiem uzdevumiem. Diskriminators ir klasifikators, kas apmācīts atšķirt demonstrāciju kopas trajektorijas no visām pārējām. Ģenerators no sistēmas stāvokļiem vai novērojumiem ģenerē darbības tā, lai diskriminators nebūtu spējīgs atšķirt tās no parauga. Formāli aprakstīt veidu, kā līdz šāda modeļa piemērotībai nonākts, ir sarežģīti, tāpat — pierādīt šāda optimizācijas procesa konverģenci. Taču intuitīvi diezgan skaidrs, ka pēc sekmīgas abu modeļu apmācības būs iegūta stratēģija, kas tuvu aproksimē ievades datu sadalījumu.

2.1.5. Uzdevumu simboliska dekompozīcija

Vēl viena metode atdarināšanas spēju uzlabošanai ir telpisku kustības trajektoriju pārvēršana simbolisku, diskrētu darbību virknē. Pamatideja ir tāda, ka vieglāk iemācīties robusti izpildīt primitīvas kustības un tad šādu primitīvo kustību secību kāda uzdevuma izpildē, nekā no neliela demonstrācija skaita iemācīties katru uzdevumu pilnībā no jauna. Šī arī nebūt nav jauna ideja — izteikta jau 1980os un 1990os gados[16]. Jau 2002. gadā veikti pētījumi par algoritmiem, kas ļauj aproksimēt trajektorijas elementus ar autonomu, nelineāru diferenciālvienādojumu sistēmām[25] un iemācīt pietiekami sarežģītas kustības — piemēram, tenisa bumbas sišanu — ar samērā nedaudziem piemēriem (ap 20). Ap to pašu laiku piedāvātas arī pieejas šādu primitīvu kustību kombinēšanai[26].

Pirms nesenā ļoti lielu neironu tīklu modeļu popularitātes uzplaukuma, šķiet, tieši simboliskās dekompozīcijas metodes bijušas starp perspektīvākajām. Robotikas literatūrā pirms 2010. gada[17] gari un plaši rakstīts par šādiem paņēmieniem, taču pēdējos gados šī popularitāte varētu būt sarukusi. Jebkurā gadījumā, aktīva pētniecība nozarē vēl joprojām notiek, it sevišķi pielietojumiem, kur robots tiek mācīts ar kinestētiskām metodēm. Piemēram, "*A Framework of Hybrid Force/Motion Skills Learning for Robots*" [27], kas publicēts 2020. gadā, šāda pieeja tiek veiksmīgi izmantota uzdevumiem, kur svarīga ne tikai telpiskā trajektorija, bet arī uz apkārtējo vidi izdarīto spēku profils (piemēram, galda tīrīšanā).

2.2. Novērojumu atdarināšana

2.2.1. Kinestētiskie novērojumi

2.2.2. Video

2.2.3. Telpas modeļi

2.3. Adaptīvu un atdarinošu metožu kombinācija

3. PRAKTISKA REALIZĀCIJA - RĪKI, PIEMĒRI

3.1. Simulācijas vides un saskarne

3.2. Vienkāršu modeļu realizācijas individuālai izpratnei

3.2.1. Stimulētā mašīnmācīšanās

3.2.2. Uzvedības kopēšana

3.2.3. DAgger

SECINĀJUMI

ATSAUCES

[1] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[2] Seppo Linnainmaa. “The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors”. In: *Master’s Thesis (in Finnish), Univ. Helsinki* (1970), pp. 6–7.

[3] Kunihiko Fukushima. “Neocognitron: A hierarchical neural network capable of visual pattern recognition”. In: *Neural networks* 1.2 (1988), pp. 119–130.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[5] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[6] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.

[7] Isaac Asimov. *I, robot*. Vol. 1. Spectra, 2004.

[8] J. Grundspeņķis. *Nacionālā enciklopēdija - mākslīgais intelekts*. 2021. URL: `https://enciklopedija.lv/skirklis/24447-m%C4%81ksl%C4%ABgais-intelekts` (visited on 01/14/2022).

[9] Beijing Academy of Artificial Intelligence. *Suggested Notation for Machine Learning*. 2020. URL: `http://ctan.math.utah.edu/ctan/tex-archive/macros/latex/contrib/mlmath/mlmath.pdf` (visited on 01/14/2022).

[10] Alexandre Attia and Sharone Dayan. “Global overview of imitation learning”. In: *arXiv preprint arXiv:1801.06503* (2018).

[11] Abhishek Gupta et al. “Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning”. In: *arXiv preprint arXiv:1910.11956* (2019).

[12] Daniel Brown et al. “Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations”. In: *International conference on machine learning*. PMLR. 2019, pp. 783–792.

[13] Peter Englert and Marc Toussaint. “Learning manipulation skills from a single demonstration”. In: *The International Journal of Robotics Research* 37.1 (2018), pp. 137–154.

[14] Richard S Sutton and Andrew G Barto. “Reinforcement learning: An introduction”. In: MIT press, 2018, pp. 60–77.

[15] Ashvin Nair et al. “Overcoming exploration in reinforcement learning with demonstrations”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 6292–6299.

[16] S Muench, J Kreuziger, and M Kaiser. “Robot programming by demonstration (rpd)-using machine learning and user interaction methods for the development of easy and comfortable robot programming systems”. In:

[17] Aude Billard et al. “Handbook of robotics chapter 59: Robot programming by demonstration”. In: *Handbook of Robotics. Springer* (2008).

[18] Alex Owen-Hill. *The Decade of Artificial Intelligence*. 2021. URL: `https://blog.robotiq.com/what-are-the-different-programming-methods-for-robots` (visited on 01/16/2022).

[19] ABB Group et al. “Special report: Robotics–ABB group”. In: *ABB Review* (2016).

[20] Dean A Pomerleau. *Alvinn: An autonomous land vehicle in a neural network*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA ARTIFICIAL INTELLIGENCE and PSYCHOLOGY . . ., 1989.

[21] Stéphane Ross, Geoffrey J Gordon, and J Andrew Bagnell. “No-regret reductions for imitation learning and structured prediction”. In: *In AISTATS*. Citeseer. 2011.

[22] Pieter Abbeel and Andrew Y Ng. “Apprenticeship learning via inverse reinforcement learning”. In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 1.

[23] Faraz Torabi, Garrett Warnell, and Peter Stone. “Generative adversarial imitation from observation”. In: *arXiv preprint arXiv:1807.06158* (2018).

[24] Jonathan Ho and Stefano Ermon. “Generative adversarial imitation learning”. In: *Advances in neural information processing systems* 29 (2016), pp. 4565–4573.

[25] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. “Movement imitation with nonlinear dynamical systems in humanoid robots”. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*. Vol. 2. IEEE. 2002, pp. 1398–1403.

[26] Stefan Schaal, Auke Ijspeert, and Aude Billard. “Computational approaches to motor learning by imitation”. In: *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 358.1431 (2003), pp. 537–547.

[27] Ning Wang, Chuize Chen, and Alessandro Di Nuovo. “A framework of hybrid force/motion skills learning for robots”. In: *IEEE Transactions on Cognitive and Developmental Systems* 13.1 (2020), pp. 162–170.