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 **3**

1.1 Darba mērķis un struktūra 4
1.2 Terminoloģijas tulkojumi 4
1.3 Tehniskās priekšzināšanas, definīcijas 5
1.3.1 Parametriski modeļi, šabloni 5
1.3.2 Neironu tīkli 7
1.3.3 Markova lēmumu procesi 7
1.3.4 Stimulētā mašīnmācīšanās 8
1.3.5 Robotikas uzdevumi 10

2 Līdzšinējie pētījumi **11**

2.1 Labi definētu trajektoriju kopēšana 11
2.1.1 Vienkāršas metodes 12
2.1.2 Statistiskas korekcijas 14
2.1.3 Inversā stimulētā mācīšanās (IRL) 15
2.1.4 Ģeneratīvie sāncenšu tīkli 15
2.1.5 Uzdevumu simboliska dekompozīcija 16
2.2 Novērojumu iegūšana, interpretācija, papildināšana 17
2.2.1 Nezināmas darbības 17
2.2.2 Dinamikas vispārināšana, atdarināšana 18
2.2.3 Demonstrācijas no cilvēka darbībām 19
2.2.4 Video demonstrācijas, perspektīvu pārbīde 20
2.2.5 Datu sintēze, telpiski modeļi 21
2.3 Atdarināšana un adaptācija, vispārināšana 22
2.3.1 Neoptimālu demonstrāciju uzlabošana 22
2.3.2 Demonstrācija — sākumpunkts apmācību procesam 23
2.3.3 Tūlītēja trajektoriju atdarināšana 23
2.3.4 Nestrukturētas demonstrācijas, plānu veidošana no galamērķiem . . 24

3 Secinājumi, rīcības plāns **27**

Atsauces **30**

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.

2. LĪDZŠINĒJIE PĒTĪJUMI

Šīs nodaļas mērķis ir izveidot aptuvenu nozares pētniecības saturisku 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.

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 pašreizējo 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 iegūšana, interpretācija, papildināšana — ne vienmēr ir pieejami dati padevīgā formā, lai tiešā veidā varētu imitēt tajos veiktās darbības. Daudzi pētījumi nodarbojas tieši ar apmācībai derīgu novērojumu iegūšanu — netiešu (piemēram, video) novērojumu pārveidošanu, labākām cilvēka-robota saskarnes metodēm, trajektorijām ar nezināmām darbībām, u.c.;
3) atdarināšana un adaptācija, vispārināšana — atdarinošās mācīšanās pielietojums, lai uzlabotu stimulēto, un otrādi; vispārīgu prasmju iegūšana no demonstrācijām, tūlītēja atdarināšana. Kā panākt, ka neaprobežojamies ar tikai piemēros esošo un spējam pielāgoties? Kā efektīvi uzsākt stimulēto mācīšanos ļoti retinātās atdevju telpās?

2.1. Labi definētu 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].

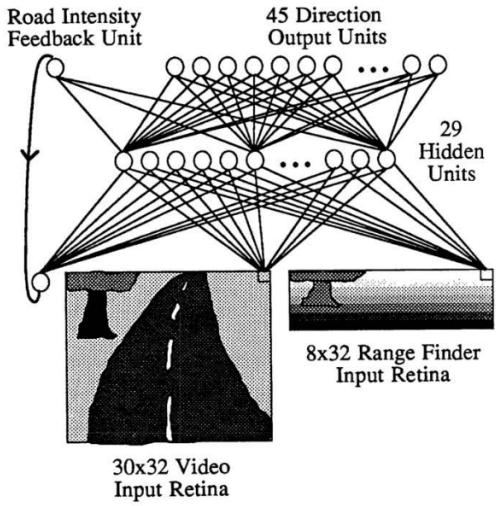
Darba autors var pats personīgi izdarīt zināmus secinājumus par tiešu trajektoriju ierakstīšanu un atkārtošanu, jo ir strādājis kā mehatronikas inženieris uzņēmumā, kas nodarbojas ar rūpnieciskās ražošanas iekārtu projektēšanu, izgatavošanu un automatizāciju, tāpēc pietiekami daudz nodarbojies arī ar 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



Att. 1: ALVINN modeļa uzbūve[20]

laikrindām, turklāt jāstrādā ir ar novērojumiem, nevis stāvokļiem. Lai veidotu vispārīgu izpratni par atdarinošo mašīnmācīšanos, noderīgi ir detalizētāk aplūkot kādu vienkāršu tās piemēru. Uzvedības klonēšana sistēmā, kur netiek veikta intensīva treniņa datu pārstrāde vai vadības algoritma argumentu pārveidošana ļoti labi kalpo šādiem mērķiem, tāpēc piemēram 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.

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

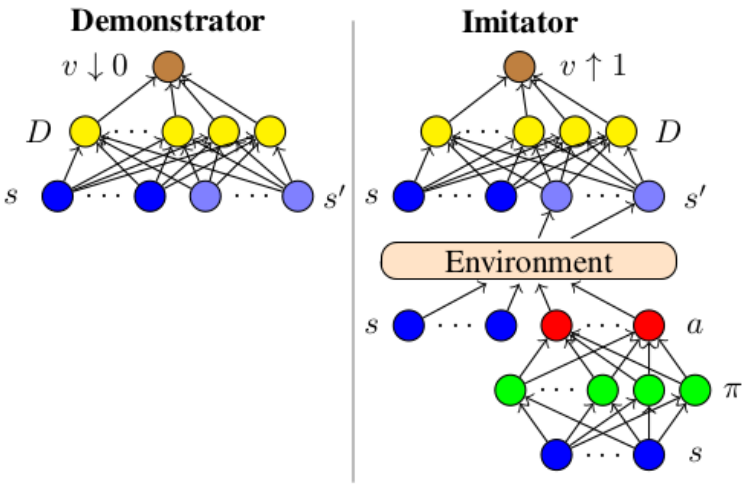
$$\mu^* = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{T} \gamma^t \phi(s_t^i) \tag{2.3}$$

Tad tiek iteratīvi atkārtota procedūra, kur atrod kādu svaru vektoru w^i un attiecīgi empīrisku atdeves funkciju $R^i(s) = w^{iT}\phi(s)$, 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



Att. 2: GAN uzbūves piemērs. Augšējais tīkls — diskriminators — cenšas atšķirt eksperta demonstrācijas no ģeneratora radītām trajektorijām[24]

trūkums IRL metodēm tiek minēta nepieciešamība katrai iteratīvi iegūtajai stratēģija no jauna veikt stimulēto apmācības procesu, lai būtu iespējams iegūt šīs metodes ģenerētas 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*"[23]. 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[24].

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 vai to elementus no visām pārējām. Ģenerators no sistēmas stāvokļiem vai novērojumiem ģenerē darbības tā, lai diskriminators nebūtu spējīgs atšķirt tās no paraugā. 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], kur mašīnmācīšanās lietojums attiecināts ne tikai uz atsevišķajiem trajektorijas primitīviem, bet arī uz katrai demonstrācijai atbilstošas to secības meklēšanu.

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 iegūšana, interpretācija, papildināšana

Pieņemot, ka pastāv robustas metodes trajektoriju ģenerēšanai, paveras cita, potenciāli daudz sarežģītāka problēma. Kā jau iepriekš minēts, jebkurā praktiskā fizikāla procesa kontroles sistēmā zināmie sistēmas momentānā stāvokļa atribūti patiesībā veido novērojumu o_t, kas ir gaistoši niecīga daļa no visiem iespējamiem, turklāt daudzkārt ir pieejami tikai šie te novērujumi — datu kopā darbības a_t tiešā veidā nav iekļautas, tās ir nepieciešams atjaunot.

Strādājot apstākļos, kur vienkārši analītiski modeļi ar labu precizitāti var paredzēt stāvokļa atribūtus un sakarības starp tiem, šī atšķirība var nebūt īpaši svarīga — piemēram, darbojoties ar robotu, kas nav sevišķi smagi noslogots un kura kontroles sistēma spēj bez lielām novirzēm atdarināt no tās prasīto kinemātiku, darboties ar novērojumiem, kuros zināmi tikai šie kinemātiskie atribūti, nav daudz sarežģītāk (varbūt pat vienkāršāk), nekā ar smalkāku situācijas aprakstu, kur pieejami arī visi dinamikas parametri.

Taču daudziem potenciāli ļoti noderīgiem lietojumiem tieši apmācībai derīgu treniņa datu kopu iegūšana no nepilnīgiem novērojumiem var būt galvenais šķērslis. Problēmas var sagādāt demonstrāciju ģenerēšana ar citādas ģeometrijas instruktoru (piemēram, cilvēku), citu objektu sarežģītu un iepriekš nezināmu konfigurāciju modelēšana (manipulējamo objektu novietojums, orientācija, u.t.t.), trajektoriju automātiska iegūšana no datu kopām, kas nav tiešā veidā paredzētas šim mērķim (video ieraksti). Tāpat jāparvar zināmi izaicinājumi, lai datus varētu ģenerēt ar netiešām metodēm — piemēram, attālinātu vadību vai virtuālās realitātes simulācijām.

2.2.1. Nezināmas darbības

Pētījumos, kuru galvenā būtība ir dažādu veidu pārveidojumi ar kinemātiskiem vai dinamiskiem trajektoriju datiem, varētu teikt, ka uzsvars ir uz kinestētiskajiem mācīšanās

aspektiem. Ja iepriekšējā nodaļā aplūkoto eksperimentu veicēji pārsvarā pieņēmuši, ka pieejami pareizi formatēti dati, kuros novērojums ir cieši sakarīgs ar robota vadībai svarīgiem sistēmas atribūtiem un zināmas katrā trajektorijas solī veiktās darbības, tad šajā tiek apskatīti gadījumi, kad šie dati ir kādā ziņā nepietiekami vai pārveidoti.

Pirmais sarežģījums, ko varētu ieviest, ir darbību trūkums demonstrācijās. Ar to jāsastopas ļoti daudzos uzdevumos — no nemarķētiem datiem var kā nebūt iegūt, piemēram, robota gala efektora pozīciju un rotāciju, taču nekas nav zināms par tā locītavu lenķiskajiem paātrinājumiem. Lai varētu izmantot jau zināmos atdarinošās mācīšanās algoritmus, rodas nepieciešamība uzminēt attiecīgās darbības, kas rezultē pārejā no viena stāvokļa uz nākamo.

"*Behavioral Cloning from Observation*"[28] (2018) ir viena šāda metode. Tās mērķis ir realizēt jau aprakstīto uzvedības klonēšanas algoritmu laikrindu datiem, kuros pieejami tikai novērojumi. Lai to panāktu, tiek ieviests papildus modelis, tikai šoreiz nevis stratēģijas, bet gan paša robota (vai cita aģenta) dinamikas aproksimēšanai.

Nedaudz vienkāršojot tad algoritma soļi ir sekojoši:

1) doto trajektoriju kopu pārveido formā $T_{dem} = \{(s_t, s_{t+1})\}$, kas ir pārejas starp stāvokļiem;
2) stratēģija π_θ un sistēmas dinamikas modelis $M_\phi(a|s, s') = P(a|s, s')$ tiek nejauši inicializēti;
3) ģenerē trajektorijas ar π_θ, pievieno trajektoriju kopai $T = \{(s_t, a_t, s_{t+1})\}$;
4) apmāca M_ϕ uz T;
5) trenē π_θ uz $\{(s_t, \max_a M_\phi(a|s, s'), s')|(s, s') \in T_{dem}\}$
6) atkārto soļus 3-5 līdz sasniegti pieņemami rezultāti

Redzams, ka pakāpeniski tiek iegūts dinamikas modelis, kas pareizi paredz pārejas funkcijas slēpto darbības parametru, un, tāpat kā parastajā gadījumā, tiek apmācīta atbilstoša stratēģija. Interesanti arī, ka šajā situācijā potenciāli nedaudz lielāks uzsvars ir tieši uz nākamo novērojumu laikrindā, nevis izvēlēto darbību — pie s, π_θ iemācās paredzēt *s' sasniegšanai labāko darbību*, nevis vienkārši atkārtot pašu darbību bez konteksta, tātad savā ziņā sistēmas dinamika un vēlamais rezultāts tiek ņemti vērā. Rezultātos autori salīdzina šo metodi ar citām, kas izmanto arī darbību datus no demonstrācijām, un konstatē, ka šis algoritms pat strādā labāk nekā tādas, ja tiek vērtēts pēc nepieciešamo demonstrācijas trajektoriju vai simulācijas iterāciju skaita.

Iepriekšējā apakšnodaļā 2. attēls ir no "*Generative Adversarial Imitation from Observation*"[28] (2018), kas turpina darboties tajā pašā virzienā, tikai šoreiz ar GAN modeļa palīdzību. Tā vietā, lai modelētu sistēmas dinamiku, diskriminators klasificē trajektoriju laikrindās sastopamās stāvokļu pārejas (s, s') pēc tā, vai tās būtu sastopamas demonstrācijā, bet ģenerators (kas reizē ir arī stratēģija π_θ) tiek trenēts, lai tā izvēlēto darbību rezultātā iegūtās stāvokļu pārejas $s = T(s, a)$ nebūtu atšķiramas no piemēriem.

2.2.2. Dinamikas vispārināšana, atdarināšana

Cita veida problēma, kas arī prasa korekciju ieviešanu, ir atdarināšana sistēmām ar mainīgu dinamiku. Robotiem izmantojot "*lead-through*" programmēšanas metodi,

kustības tiek programmētas bez slodzes un, iespējams, neievērojot ātrumu — zināma tikai daļēja sistēmas kinemātika. Ja var paredzēt, ka pēc tam ekspluatācijā atšķirsies robotu slogojošie spēki un griezes momenti, tad var meklēt veidus, kā šīs novirzes jau laicīgi kompensēt. "*Online Movement Adaptation based on Previous Sensor Experiences*"[29] (2011) paredz jau iepriekš apraksītīto simboliskās dekompozīcijas modeļu papildināšanu ar korekcijām reālā laikā, izmantojot atgriezenisko saiti ar gan paša robota iekšējiem devējiem, gan ārējiem sensoriem.

Pētījumos, kas nodarbojas ar simbolisko dekompozīciju, bieži vien tiek aprakstīti visai sarežģīti un tieši robotu dinamikai specifiski matemātiskie modeļi, taču vienkāršoti procesu var aprakstīt sekojoši: kustības rakstorojošie diferenciālvienādojumu modeļi tiek iegūti, robotu manuāli vai citādi pārvietojot. Tad kustība tiek veikta ar pareizu kinemātiku (ātrumiem, paātrinājumiem), bet bez papildu slodzes. Tiek ierakstīti sensoru novērojumi un veidoti kustībai atbilstoši modeļi, kas paredz šos raksturlielumus trajektorijas gaitā. Autoru vārdiem — robots iemācās, kā kustībai vajadzētu "justies". Tad reāli ekspluatācijā var sekot līdzi nobīdēm no normas un ar klasiskās kontroles teorijas metodēm veikt korekcijas.

Nodarbojoties ar ļoti sarežģītiem uzdevumiem — piemēram, salikšanas procesiem — mēģināt vadīt robotu cauri visiem iespējamiem detaļu savstarpējiem stāvokļiem var būt neiespējami. Ja var iegūt demonstrācijas kādā citā, vieglāk realizējamā veidā un izstrādāt vispārīgu metodi, kā tos atdarināt neatkarīgi no robota uzbūves, rodas iespējas automatizēt arī šādus procesus. "*Contact Skill Imitation Learning for Robot-Independent Assembly Programming*"[30] (2019) realizē šādu procedūru, izmantojot divus būtiskus elementus:

1) *forward dynamics compliance control* — robota vadības algoritmu, kurā tiek kontrolēti uz efektoru iedarbojošos spēku un griezes momentu vektori, nevis izpildelementu pozīcija tiešā veidā[31];
2) reukurentos neironu tīklus laikrindas nākamā elementa paredzēšanai — t.i., tīklus kuru ievadē parādās iepriekšējā laika soļa tā paša modeļa izvades vektors;

Demonstrāciju iegūšanai cilvēks simulācijas vidē ar datorpeles palīdzību veic salikšanas procesu, izmantojot tikai vizuālo uztveri un intuitīvu izpratni par sadursmju dinamiku. Novērojumus veido manipulējamā objekta masas centrā reģistrēto griezes momentu un spēku vektori. Stratēģija — rekurentais neironu tīkls — π_θ tiek apmācīta paredzēt nākamo spēku/momentu vektoru laikrindā, un tas tiek izmantots robota kontrolei ar minēto vadības algoritmu.

2.2.3. Demonstrācijas no cilvēka darbībām

Ja nepieciešams ātri un lēti radīt treniņa datu kopas apmācības procesiem — kā tas noteikti būtu jebkurai praktiski izmantojamai robotu programmēšanas sistēmai — svarīgi radīt cilvēkam draudzīgu saskarni. Tā vietā lai programmētu robota trajektoriju ar tradicionālām metodēm, ir veikti mēģinājumi attīstīt metodes, kas ļautu dabiski ierakstīt cilvēka izpildītas kustības un izteikt tās formā, ko var izmantot robota apmācībai. Viens no virzieniem, kurā veikta izpēte, ir tieša fiziskās kinemātikas ierakstīšana un pārveidošana

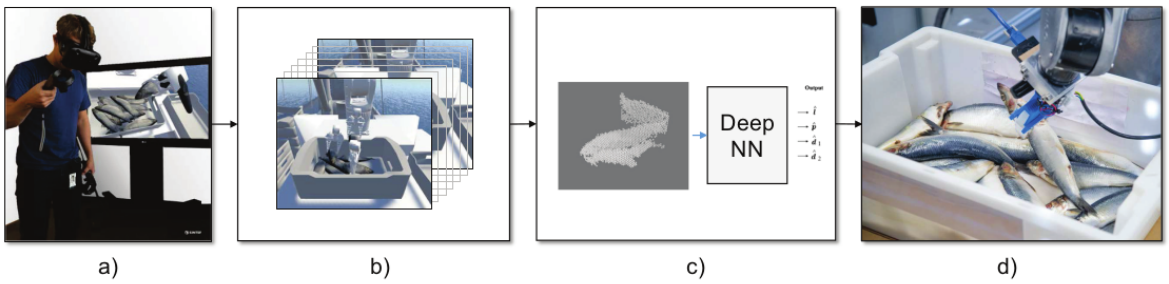
— "*Imitation learning in industrial robots: a kinematics based trajectory generation framework*"[32] (2017) ir ilustratīvs piemērs. Tiek izmantota *Microsoft Kinect* — savulaik populāra videospēļu vadībai paredzēta kustību uztveres ierīce, kas spēj ierakstīt kāda objekta kustību aprakstošu punktu virkni telpā — lai ierakstītu cilvēka kustības. Tad ar analītiskām metodēm tiek veikta trajektoriju interpolācija, grupēšana ar klasterizācijas algoritmu, un izpildē vēlamo trajektoriju iegūst ar tuvāko kaimiņu metodēm.

Lai mazinātu atšķirības starp cilvēkam un robotam pieejamām novērojumu un darbību telpām, var izmantot virtuālo realitāti — gan simulācijās, gan robota tālvadībai. "*Deep imitation learning for complex manipulation tasks from virtual reality teleoperation*"[33] (2018) ir aprakstīta fiziska robota tālvadības sistēma, kas robota novēroto vidi pārveido sintētiskā telpā, kur cilvēks var ar manipulatoru palīdzību intuitīvi vadīt robota kustības. Šis uzdevums nav gluži triviāls, jo jāpārvar atšķirības starp, piemēram, robota kameras un cilvēka galvas kustību ātrumu, lai neradītu diskomfortu lietotājam. Tādējādi iegūta demonstrāciju datu kopa, kas sastāv no attēliem un telpiskiem dziļumiem, ko arī tālāk var izmantot konvolūciju neironu tīklā, lai realizētu uzvedības klonēšanu.

2.2.4. Video demonstrācijas, perspektīvu pārbīde

Ļoti plašas iespējas demonstrāciju iegūšanai pavērtu spēja tās iegūt no video datiem un izmantot šādas "redzes" sistēmas arī tiešā vadības uzdevuma risināšanai. Izrādās, ka daži no jau iepriekš aplūkotajiem algoritmiem ir pietiekami vispārīgi, lai tos varētu pielāgot šim uzdevumam. Piemēram, GAN no novērojumiem[24] ir ļoti robusts attiecībā pret ievades datu formu un saturu, ja vien tajos ir iespējams pietiekami labi atšķirt demonstrāciju no citām trajektorijām. Tāpat kā gadījumā, kad strādā ar kinemātiku aprakstošiem stāvokļiem, GAN vienkārši tiek apmācīts diskriminēt/ģenerēt trajektoriju vizuā-lās reprezentācijas, ar samērā labiem rezultātiem.

Brīdī, kad vairs netiek izmantoti robota konfigurāciju aprakstoši novērojumi, rodas jauna problēma — iegūtie novērojumi ir atkarīgi no izmantotās perspektīvas, kas vispārīgā gadījumā var arī nesakrist ar robota vadības algoritmu ievades datus ģenerējošo. "*Imitation from observation: Learning to imitate behaviors from raw video via context translation*"[34] (2018) risina šo atšķirīgo kontekstu sarežģījumu un iegūtu modeli izmanto arī rezultātu uzlabošanai vispār. Pieņemot, ka trajektorijai atbilstošie novērojumi tiek iegūti no dažādām perspektīvām telpā, vispirms tiek apmācīts enkodera-dekodera tipa neironu tīkls, kas iegūst novērojumu vektoriālas reprezentācijas, ko pēc tam iespējams dekodēt par tai pašai sistēmas konfigurācijai atbilstošu novērojumu no jebkuras perspektīvas. Tas ļauj ne tikai tiešā veidā "tulkot" novērojumus no vienas kameras uz citu, bet arī izmantot iegūtās kodētās reprezentācijas Eiklīda distanci no demonstrāciju trajektoriju soļiem atbilstošajām kā atdeves funkciju stimulētās mašīnmācīšanās algoritmam, jo arī intuitīvi skaidrs, ka modelis, kas spēs atjaunot situācijas attēlu kādā perspektīvā, ja zināmi tikai no citas perspektīvas gūtie attēli, kaut kādā ziņā sevī ietver visus nozīmīgos konfigurācijas atribūtus.



Att. 3: Zivs satveršanas modeļa apmācība: a) cilvēka radīto demonstrāciju iegūšana; b) datu kopas sintētiska pavairošana; c) telpiska konvolucionālā neironu tīkla apmācība; d) veiksmīga zivs satveršana realitātē.[35]

2.2.5. Datu sintēze, telpiski modeļi

Daudzi uzdevumi, kam tiek izmantoti roboti, sevī ietver manipulāciju ar citiem objektiem, kuru telpisko novietojumu un citus atribūtus ne viemēr viegli iegūt, un nākas paļauties uz netiešiem novērojumiem, ko sagādā, piemēram, tehniskās redzes sistēmas. Ja objekti ir paredzamā orientācijā un regulāras formas, no šīs problēmas parasti iespējams izvairīties, taču vēl joprojām daudzi pārvietošanas un pozicionēšanas procesi tiek veikti ar cilvēka roku darbu.

"*Teaching a robot to grasp real fish by imitation learning from a human supervisor in virtual reality*"[35] ir visnotaļ interesants un praktisks projekts, kura ietvaros izstrādāts modelis zivju satveršanas uzdevumam. Tajā robotam ir jāspēj no kastes, kurā atrodas vairākas zivis, satvert un pārvietot vienu. Tā kā zivs ir ļoti neregulāras formas objekts, turklāt slidens un padevīgs satveršanai ar robota efektoru tikai ļoti ierobežotā iespējamo kontakta konfigurāciju apakškopā, šis ir samērā ambiciozs pētījums.

Risinājuma pamatā ir trīs idejas:

1) zivs pozīcijai un orientācijai piemērota satveršanas punkta un lenķa paredzēšana no ar telpisko kameru gūtiem telpiskiem punktu mākoņa datiem, izmantojot konvolucionālo neironu tīklu — izmantojot vispārinātas metodes, kas sākumā izmantotas divdimensionāla attēlu klasifikācijai;
2) virtuālās realitātes vidi, kas ļauj cilvēkam viegli un dabiski ģenerēt demonstrācijas;
3) datu kopas sintētisku pavairošanu.

Novērojumi gūti, cilvēkam VR vidē brīvā veidā satverot zivis dažādos veidos. Lai viennozīmīgi aprakstītu katru iespējamo satvērumu, izmantoti trīs vektori — pozīcija (punkta), garenās ass orientācija un rotācija ap to. Pēc tam ģenerēts liels skaits zivju izvietojuma konfigurāciju. Lai sintētiskie dati būtu lietojami, nepieciešams nodrošināt, ka satvēruma matrica tiek koriģēta atbilstoši zivs izliekumam, novietojumam un orientācijai. Katrā sagatavotajā sistēmas konfigurācijā katrai zivij sākumā piešķirti visi iespējamie satvēruma vektori, no kuriem atsijāti visi, kas kolīziju dēļ nav sasniedzami. Šie dati tad izmantoti 3-dimensionālā konvolūciju neironu tīkla apmācībai — simulatorā ģenerēti ainai atbilstoši telpiskie attēli, modelis apmācīts kā tipisks klasifikators. Rezultātā konstatēts, ka aptuveni 74% izdarīto satveršanas mēģinājumu bijuši sekmīgi.

2.3. Atdarināšana un adaptācija, vispārināšana

Ja pieejamas demonstrācijas no eksperta, to atdarināšana tiešā veidā varbūt ir pirmais solis noderīgu stratēģiju iegūšanā, taču nebūt ne vienīgais, ko iespējams darīt. Līdz šim aplūkotas metodes, kas izrāda zināmu adaptācijas pakāpi, lai precīzāk un robustāk imitētu paraugus, taču vienmēr izdarīts pieņēmums, ka instruktors kādā ziņā optimāli izpildījis uzdevumu. Turklāt visos gadījumos, lai apmācītu sistēmu izpildīt jaunu uzdevumu, nepieciešams veikt intensīvu treniņu ar cilvēka starpniecību.

Šajā apakšnodaļā aprakstīti pētījumi, kuros meklēti veidi, kā apvienot atdarināšanu ar adaptāciju, lai vispārinātu demonstrācijās ietverto informāciju, atvieglotu jaunu uzdevumu programmēšanas procesu vai pārspētu instruktoru sasniegto rezultātu kvalitātē. Lai gan mēģinājumi darboties šajā jomā nav nekas jauns — var atrast senākus pētījumus, kuros, piemēram, izmantotas sarežģītas un tieši robotikas mērķiem specifiskas simboliskās dekompozīcijas metodes[36] — uzsvars šeit tiek likts uz nesenākām un vispārīgākām metodēm, kas attīstītas jau pašreizējā neironu tīklu uzplaukuma periodā.

2.3.1. Neoptimālu demonstrāciju uzlabošana

Varētu sacīt, ka jau pati atdarinošās mācīšanās pamatnostādne — apgūt stratēģiju problēmas risināšanai, pēc iespējas tuvāk sekojot kādai demonstrāciju kopai — sevī ietver pieņēmumu par demonstrācijas optimalitāti. Taču izrādās, ka iespējams pašu treniņa datu kopu izmantot, lai gūtu informāciju par to, ko instruktors patiesi vēlējies sasniegt, un attiecīgi optimizēt modeli šī slēptā mērķa sasniegšanai.

"*Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations*"[12] (2019) izmanto jau iepriekš aprakstīto inversās stimulētās mācīšanās algoritmu saimi par pamatu jaunai metodei. Tā vietā, lai atrastu atalgojuma funkciju, kas pamato demonstrāciju kopas trajektorijas τ, tiek meklēta tāda, kas skaidro to *sakārtojumu*. Tāpēc nepieciešāms datu kopu papildināt ar kārtojumu $\prec$, lai

$$\tau_i \prec \tau_j \Rightarrow \sum_{s_t \in \tau_i} R(s) \leq \sum_{s_t \in \tau_j} R(s) \tag{2.4}$$

Protams, kārtojums var arī nebūt ideāls — trajektoriju vērtējums var būt subjektīvs vai trokšņains, ja nav zināma īstā atalgojuma funkcija, tāpēc jārēķinās ar kļūdainu pāru attiecību pastāvēšanas iespējamību ar kļūdas varbūtību ϵ

$$\exists \epsilon > 0 : \tau_i \prec \tau_j \Rightarrow P\left(\sum_{s_t \in \tau_i} R(s) \leq \sum_{s_t \in \tau_j} R(s)\right) \geq 1 - \epsilon \tag{2.5}$$

Ja kārtojums ir pieejams, radoši nosauktais *Trajectory-ranked Reward EXtrapolation* jeb T-REX algoritms darbojas divos soļos:

1) izmantojot demonstrāciju datus, tiek apmācīta nezināmās atalgojuma funkcijas $R(s)$ aproksimācija $r_\phi(s)$ — parametrisks modelis, konkrēti neironu tīkls;
2) tāpat kā IRL gadījumā, rekonstruētā atalgojuma funkcija tiek izmantota, lai apmācītu stratēģiju π_θ ar stimulētās mašīnmācīšanās metodēm.

Eksperimentāli pārbaudīts, ka piemēros, kur zināmas demonstrācijām atbilstošās īstās atalgojuma funkcijas un kļūdas varbūtība saglabājas zem 15%, rekonstruētā atalgojuma funkcija r_ϕ daudzos gadījumos labi aproksimē īsto. Turklāt, ja izmantotās demonstrācijas nav optimālas, iegūtā stratēģija π_θ tās pārspēj.

2.3.2. Demonstrācija — sākumpunkts apmācību procesam

Atšķirībā no IRL un augstāk aprakstītā T-REX, ir iespējams uz atdarinošās un stimulētās mācīšanās kombināciju skatīties arī no otras puses — nevis izmantot stimulētās metodes, lai realizētu atdarināšanu, bet gan izmantot demonstrācijas, lai uzlabotu parasto stimulētās mašīnmācīšanās procesu ar zināmu atalgojuma funkciju.

"*Deep Q-learning from demonstrations*"[37] (2018) piedāvā vispārīgu metodi stimulētās mācīšanās procesa inicializācijai ar demonstrāciju datu kopas palīdzību. Šādu pieeju motivē viens no lielākajiem klupšanas akmeņiem RL algoritmu apmācībā — ja atalgojuma funkcijas nenulles vērtības konfigurāciju telpā ir stipri retinātas, kā tas ļoti bieži ir arī dažādos robotikas uzdevumos, tad, praktiski apmācot stratēģiju izpildīt tai uzdoto uzdevumu, nepieciešams ļoti liels skaits treniņa soļu — lai šo telpu apstaigātu, atrastu vēlamos rezultātus un optimizētu trajektoriju caur tai.

Q-mācīšanās gadījumā stratēģiju var uzdot formā $\pi^{\epsilon Q_\theta}$, kur

$$T(s,a) = P(s') \to Q_\theta(s,a) \sim \mathbb{E}[v(s')] \tag{2.6}$$

jeb modelis Q_θ aproksimē kopējā atalgojuma sadalījumu pār iespējamām darbībām (jeb nākamo stāvokļu vērtības), un attiecīgi izvēlas vienu pēc "ϵ-alkatīgas" stratēģijas

$$a = \underset{a}{\operatorname{argmax}}\, Q(s,a) \text{ ar varbūtību } 1-\epsilon;\ \text{nejauši izvēlēta citādi} \tag{2.7}$$

Demonstrācijas tad tiek izmantotas, lai apmācītu Q_θ vēl pirms notiek jebkāda modeļa mijiedarbība ar vidi, un attiecīgi tas jau uzreiz darbojas daudz kvalitatīvāk nekā nejauši inicializēts modelis. Rezultāti liecina, ka jau sākotnējās iterācijās šāds algoritms var darboties visai labi, kas paver iespējas to izmantot situācijās, kad nav iespējams veikt apmācību simulācijas vidē, bet ir pieejami demonstrāciju dati. Robotikas kontekstā tas nozīmē, ka, ja jāveic apmācība uz fiziska robota, modeļa sagatavošana ar atdarināšanu varētu būt visai noderīgs paņēmiens.

2.3.3. Tūlītēja trajektoriju atdarināšana

Daudzas līdz šim aplūkotās atdarinošās metodes darbojas ar pieņēmumu, ka mērķis ir no demonstrāciju datu kopas iegūt modeli, kas pēc apmācības spēj izpildīt vienu uzdevumu, un attiecīgi treniņa algoritms neparedz iespēju bez papildus iterācijām atdarināt iepriekš neredzētas demonstrācijas — tipiski pieejams tikai pašreizējais sistēmas stāvoklis ievadē un vēlamā darbība izvadē.

Salīdzinot ar bioloģiskām sistēmām, uzreiz ir skaidrs, ka šādi procesi nekad nespēs izdarīt cilvēkam šķietami pašsaprotamo — novērot darbību un uzreiz to atkārtot bez liela skaita treniņa iterāciju. Lai būtu iespējams sasniegt šādu rezultātu, ir nepieciešams nevis modelis, kas ar paraugiem ticis optimizēts vienas stratēģijas atdarināšanai, bet gan tāds,

kas vispārina pašu atdarināšanas procesu — ievadē sistēmas pašreizējais stāvoklis tiek apvienots ar demonstrāciju, lai iegūtu vēlamo darbību.

"*One-Shot Imitation Learning*"[38] (2017) šādu rezultātu sasniedz, apmācot sarežģītas uzbūves modeli, kas apmācīts no vienas demonstrācijas paredzēt citas vispārīgā veidā — tā ievades vektors satur veselu treniņa kopas trajektoriju. Modelis tiek apmācīts uzreiz veselai līdzīgu uzdevumu kopai, nevis tikai vienam konkrētam.

Konkrēti, pētītā uzdevumu saime ir kuba formas detaļu kraušana vienai uz otras ar robotu. Novērojumu ģenerēšanas un interpretēšans problēmas gan tiek apietas, jo sistēmas stāvokli raksturojošie vektori satur gan robota iekšējo konfigurāciju, gan detaļu relatīvās pozīcijas attiecībā pret robota efektoru. Tad, izmantojot konvolūciju un uzmanības mehānismus, dažādu garumu trajektoriju laikrindas tiek reduētas uz vienu sistēmas stāvokļa vektoru, kas paredz piemērotāko darbību attiecībā pret doto demonstrāciju un momentāno sistēmas stāvokli.

2.3.4. Nestrukturētas demonstrācijas, plānu veidošana no galamērķiem

Droši vien lielākā atkāpe no "klasiskā" atdarinošā mašīnmācīšanās uzdevuma ir tajos pētījumos, kur tiek atmests pieņēmums, ka ir dotas diskrētas parauga trajektorijas ar zināmu uzdevumu. Tā vietā pat pārraudzītā trajektoriju marķēšanas stadija tiek automatizēta un atstāta apmācības algoritma pārziņā. Tā vietā var novērot, ka, ja mērķis ir iemācīties modeli, kas spēj vispārīgi atrast ceļu no vienas sistēmas konfigurācijas uz citu, pietiek atrast trajektorijas, kas iet caur abām. "*Learning Latent Plans from Play*"[39] (2019) ierosina demonstrāciju strukturētas ievākšanas vietā izmantot cilvēka dabisko tendenci spēlēties ar dažādiem objektiem, lai virtuālās realitātes vidē ģenerētu demonstrāciju kopas. Sistēma šajā gadījumā ir virtuālā realitātē modelēta vide, bet pieejamie novērojumi — robota iekšējā konfigurācija un simulēti attēli.

Tiek konstatēts, ka, pilnīgi nejauši ģenerējot trajektorijas, paies visnotaļ ievērojams laiks, līdz tiks apstaigāts pietiekami plašs konfigurāciju telpas reģions, lai gūtu labas trajektorijas starp jebkuriem punktiem tajās. Atšķirībā no nejauša procesa, cilvēks jau nāk ar sagatavotu izpratni par tāda tipa uzdevumu risināšanas elementiem, kas pēc tam varētu būt interesanti citiem. Brīvi pētot dažādās simulācijas vidē pieejamās iespējas, tiks dabiski izmantotas priekšzināšanas par dažādām fizikālām sakarībām starp objektiem un to mijiedarbību — piemēram, rokturu satveršanu, lai atvērtu un aizvērtu durvis, pogu nospiešanu, objektu satveršanu un pacelšanu.

Lai no šādas nestrukturētas informācijas iegūtu praktiski izmantojamas stratēģijas, nepieciešams papildināt stratēģijas formālismu ar mērķa jēdzienu — ne vairs vienkārši atrodot katram stāvoklim piekārtotu $\pi(s)$, bet gan parametrizētu pēc vēlamā beigu stāvokļa (*goal*) — $\pi(s, s_g)$. Ja pieejamas trajektoriju laikrindas formā $s_1, s_2, ..., s_n$, nav grūti pamanīt, ka jebkura apakšvirkne veido sākuma-gala stāvokļu pāri ar visām to starpā esošajām pārejām.

Tiek piedāvāti divi dažādi modeļu šabloni un apmācības algoritmi, kas spētu iegūt reālas stratēģijas no nestrukturētu datu kopas. Vienkāršojot — ignorējot sistēmas stāvokļa reprezentāciju kodēšanas detaļas — pirmo var aprakstīt samērā vienkārši. Modelis

$\pi_\theta(s, s_g)$ saņem ievadē pašreizējo un vēlamo stāvokli, un paredz nepieciešamo darbību. Kā šablons tiek izmantots rekurentais neironu tīkls, treniņa datus veido trajektorijas — pēdējais trajektorijas stāvoklis kļūst par s_g, un π_θ optimizē, lai katram (s_t, a_t, s_g) būtu $\pi_\theta(s_t, s_g) \approx a_t$. Taču pastāv problēma — var pastāvēt dažādas trajektorijas starp jebkuriem s, s_g, kas draud apgrūtināt mācīšanās procesu.

Otrs, sarežģītākais modelis apkaro šo parādību, modelējot plāna jēdzienu — tiek pieņemts, ka katrai trajektorijai τ var piekārtot rīcības plānu, visiem iespējamiem rīcības plāniem pastāv vektoriālas reprezentācijas z, un vieglāk ir vispirms paredzēt atbilstošāko plānu, pēc tam — piemeklēt tam viennozīmīgi piekārtotu trajektoriju. Modelis tad sastāv no trim daļām:

1) plānu enkodera $q_\phi(z|\tau)$, kas izsaka trajektorijas ar matemātisko cerību un standartnovirzi vektoriem $\mu_\phi, \sigma_\phi = q_\phi(\tau)$;
2) plānu selektora $q_\psi(z|s, s_g)$, kas analoģiski paredz $\mu_\psi, \sigma_\psi = q_\psi(s, s_g)$ un iegūst no sadalījuma konkrētu vektoru z diferencējamā veidā (pieskaitot vidējām vērtībām gadījuma lieluma reizinājumu ar standartnovirzi);
3) dekodera $\pi_\theta(z, s, s_g)$, kas rekonstruē nepieciešamo darbību no plāna reprezentācijas, galamērķa un sistēmas momentānā stāvokļa.

Divi varbūtību sadalījumi q nepieciešami, jo optimizācijas procesā tiek minimizēta Kulbaka-Leiblera diverģence starp tiem — ļaujot visu trajektoriju aizstāt ar tikai diviem tās punktiem. Otrs mērķa funkcijas faktors ir kļūda darbībā a_t. Tā kā visi slāņi ir diferencējami, viss modelis kopā veido vienu neironu tīklu, ko kopā arī apmāca uz trajektoriju apakšvirknēm. Stratēģija ir $\pi_\theta(q_\psi(s, s_g), s, s_g)$.

Tātad, lai programmētu robotu izpildīt kādu konkrētu uzdevumu, vairs nav nepieciešamas ne papildus treniņa iterācijas, ne demonstrācijas — pietiek norādīt vēlamo sistēmas gala stāvokli, un stratēģija to pati sasniegs. Rezultātu sekcijā secināts, ka šāda mācīšanās spēj pārspēt klasisko uzvedības klonēšanas metodi 18 dažādiem uzdevumiem — kaut gan otra izmantojusi demonstrācijas, kas speciāli sagatavotas katram. Turklāt, pateicoties faktam, ka modelis apgūst trajektorijas no katra konfigurāciju telpas punkta uz katru citu, tā ir ļoti robusta pret nobīdēm. Visbeidzot, ar dimensiju redukcijas metodēm aplūkojot iegūto rīcības plānu reprezentācijas telpu redzams, ka pēc cilvēkam saprotama mērķa līdzīgas darbības veido klasterus arī šajās projekcijās.

Interesants turpinājums šim pētījumam ir "*Language conditioned imitation learning over unstructured data*"[40], kur iegūtais rezultāts atkal pagriezts otrādi — ja ir metode, kas spēj pati izdalīt uzdevumus un grupēt tos pēc būtības, tad apvienojot to ar sagatavotiem marķējumiem, var iegūt stratēģiju, kas vadāma ar šādiem marķējumiem. Konkrēti, ja daļai no trajektorijām tiek *a posteriori* piekārtots dabiskā valodā izteikts mērķis, var apmācīt modeli, kas sasaista rīcības plānu reprezentācijas ar frāžu reprezentācijām — ļaujot vadīt robotu ar jau iepriekš apmācītu valodas enkoderu ģenerētiem kodiem. Praktiski tas izpaužas tā, ka iespējams robotam dot pavisam dabiskas komandas, turklāt neatkarīgi no izvēlētajiem sinonīmiem un pat dažādās valodās.

Ar tīri atdarinošām metodēm iegūtas stratēģijas var nebūt optimālas, it sevišķi

gadījumos, kad tiek prasīts sasniegt mērķi, kam nav sagatavota speciāla demonstrāciju datu kopa.'*Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning*"[11] (2019) piedāvā apvienot nestrukturētas demonstrāciju datu kopas un galamērķus kā uzdevuma specifikāciju ar zināmu atalgojuma funkciju. Tā varētu iegūt labas inicializācijas apmācības procesiem, kas citādi būtu nepraktiski vai teju neiespējami ar vienkāršām konfigurāciju telpas pārstaigāšanas metodēm retināto atalgojuma vērtību dēļ.

Pamatā risinājumam ir hierarhiskā stimulētā mācīšanās — tiek iegūta nevis viena stratēģija, bet divas. π_θ^h jeb augsta līmeņa stratēģija ievadē saņem pašreizējo sistēmas stāvokli s un gala stāvokli s_g, bet tā vietā, lai izvadītu vēlamo darbību, rezultātā izdod lokālu mērķi s_g^l — argumentu zemākā līmeņa stratēģijai π_ϕ^l. Otrā tad atkarībā no s, s_g^l ģenerē vēlamo darbību vidē a. Apmācība tiek realizēta divos soļos:

1) atdarināšana — katrai no stratēģijām tiek sagatavota atsevišķa datu kopa, izmantojot "slīdošā loga" principu pār trajektorijām demonstrāciju datos. π_θ^h izvēlas plašāku logu W_h un minimālo distanci līdz lokālajam mērķim j. Katram s_t tad tiek izveidoti korteži (s_t, a'_t, s_g) kur $s_g \in \{s_{t+1}, s_{t+2}, ..., s_{t+W_h}\}$ bet $a'_t = s_{t+j}$. Zema līmeņa stratēģijai π_ϕ^l tad izvēlas īsāku logu W_l un analoģiski katru tālāk logā esošo s_{t+i} izmanto kā galamērķi s_g^l, lai izveidotu (s_t, a_t, s_g^l). Šādi pārveidotos datus tad izmanto, lai apmācītu modeļus π_θ^h, π_ϕ^l atdarināt datu kopu;
2) uzlabošana — izmantojot zināmu atalgojuma funkciju $r(s, a, s_g)$, realizē stimulēto mācīšanos. Iegūtās stratēģijas tiek darbinātas vidē, iegūtie dati tiek tāpat pārkārtoti un π_θ^h, π_ϕ^l atkal tiek optimizēti, tikai šoreiz gradientu optimizācijas mērķa funkcija (*loss*) tiek papildināta ar atalgojuma faktoru.

Tiek iegūts algoritms, kas spēj no nestrukturētas datu kopas iemācīties, kā sasniegt patvaļīgus galamērķus, bet pēc tam — uzlabot iegūtās stratēģijas ar stimulētās mācīšanās palīdzību. To parāda rezultāti — dažādos uzdevumos šāda pieeja pārspēj iepriekš aprakstīto latento plānu pieeju[39], taču attiecīgi tai ir nepieciešama papildu patstāvīgas mācīšanās etaps.

3. SECINĀJUMI, RĪCĪBAS PLĀNS

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

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

[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.

[28] Faraz Torabi, Garrett Warnell, and Peter Stone. “Behavioral cloning from observation”. In: *arXiv preprint arXiv:1805.01954* (2018).

[29] Peter Pastor et al. “Online movement adaptation based on previous sensor experiences”. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 365–371.

[30] Stefan Scherzinger, Arne Roennau, and Rüdiger Dillmann. “Contact skill imitation learning for robot-independent assembly programming”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 4309–4316.

[31] Stefan Scherzinger, Arne Roennau, and Rüdiger Dillmann. “Forward dynamics compliance control (FDCC): A new approach to cartesian compliance for robotic manipulators”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 4568–4575.

[32] Abhishek Jha et al. “Imitation learning in industrial robots: a kinematics based trajectory generation framework”. In: *Proceedings of the Advances in Robotics*. 2017, pp. 1–6.

[33] Tianhao Zhang et al. “Deep imitation learning for complex manipulation tasks from virtual reality teleoperation”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 5628–5635.

[34] YuXuan Liu et al. “Imitation from observation: Learning to imitate behaviors from raw video via context translation”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1118–1125.

[35] Jonatan S Dyrstad et al. “Teaching a robot to grasp real fish by imitation learning from a human supervisor in virtual reality”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 7185–7192.

[36] Peter Pastor et al. “Skill learning and task outcome prediction for manipulation”. In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 3828–3834.

[37] Todd Hester et al. “Deep q-learning from demonstrations”. In: *Thirty-second AAAI conference on artificial intelligence*. 2018.

[38] Yan Duan et al. “One-shot imitation learning”. In: *arXiv preprint arXiv:1703.07326* (2017).

[39] Corey Lynch et al. “Learning latent plans from play”. In: *Conference on Robot Learning*. PMLR. 2020, pp. 1113–1132.

[40] Corey Lynch and Pierre Sermanet. “Language conditioned imitation learning over unstructured data”. In: *Proceedings of Robotics: Science and Systems. doi* 10 (2021).