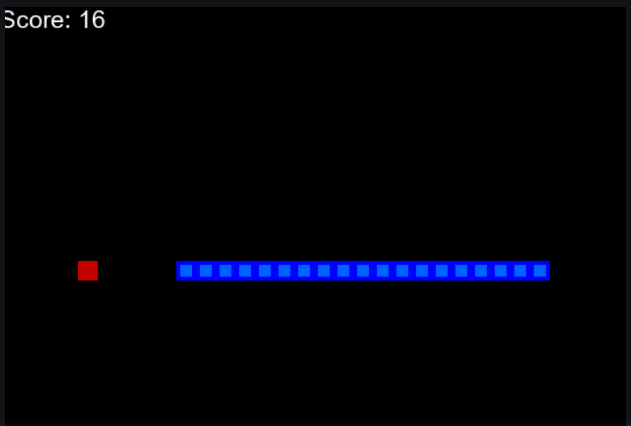
Mākslīgā intelekta mācīšana spēlēt čūsku

*Autors: Andris Karpenko*

*J.Pilsudska Daugavpils Valsts poļu ģimnāzija*

*Darba vadītājs: Antons Macuks, programmēšanas skolotājs*

*12. klase*



20.03.2023

Problēmas izpēte un analīze………………………………………………………………………3

Pētījuma metodes izvēle un pamatojums

Pētījuma procesa apraksts

Pētījuma datu kopsavilkums

Programmatūras prasību specifikācija……………………………………………………………4

Risinājuma mērķauditorijas izvēle

Programmatūras produkta un tā funkciju apraksts

Programmatūras produkta skice

Programmatūras izstrādes plāns………………………………………………………………5 - 6

Pārskats par pastiprinājuma mācīšanos Snake spēlē

Trīs moduļi: Vide, modelis, aģents

Algoritms Snake spēlei ar dziļo Q mācīšanos

Kā darbojas modeļi

Lietotāja rokasgrāmata……………………………………………………………………………..6

Kā instalēt un palaist spēli

Piemērotās licences pamatojums………………………………………………………………….7

**2**

**1. Problēmas izpēte un analīze – izpētes metodes izvēle un pamatojums, izpētes procesa apraksts, izpētes datu apkopojums.**

Cik efektīva ir Deep Q Learning algoritma izmantošana, izstrādājot mākslīgā intelekta vadītu “Čūska” spēli, un kā to var salīdzināt ar tradicionālajām spēļu izstrādes metodēm?

Pētāmā problēma ir Deep Q Learning algoritma efektivitāte mākslīgā intelekta vadītas spēles "Snake" izstrādē un tās salīdzinājums ar tradicionālajām spēļu izstrādes metodēm.

Šai problēmai izvēlētā pētījuma metode ir literatūras apskata un eksperimentālā pētījuma kombinācija. Literatūras pārskatā tika pētīti esošie pētījumi un raksti par pastiprināto mācīšanos, Deep Q Learning un spēļu izstrādi. Eksperimentālajā pētījumā tika ieviests Deep Q Learning algoritms spēles Snake izstrādē un salīdzināts ar tradicionālajām spēļu izstrādes metodēm.

Pētījuma process ietvēra šādus posmus:

Problēmas un pētījuma jautājumu definēšana

Literatūras apskats, lai apkopotu attiecīgo informāciju un atziņas.

Eksperimentālās iekārtas, tostarp spēles izstrādes procesa, datu vākšanas metožu un veiktspējas rādītāju, izstrāde.

Deep Q Learning algoritma un tradicionālo spēļu izstrādes metožu ieviešana.

Datu vākšana un analīze par abu metožu veiktspēju

secinājumu izdarīšana un ieteikumu sniegšana, pamatojoties uz iegūtajiem rezultātiem

Pētījuma dati parādīja, ka Deep Q Learning algoritms bija efektīvs mākslīgā intelekta vadītas spēles "Snake" izstrādē. Mākslīgā intelekta vadītā čūska spēja mācīties un laika gaitā uzlabot savu sniegumu, sasniedzot augstākus rezultātus nekā ar tradicionālo spēles izstrādes metodi. Tomēr izstrādes process bija sarežģītāks un laikietilpīgāks salīdzinājumā ar tradicionālajām spēļu izstrādes metodēm. Turklāt Deep Q Learning algoritms prasīja ievērojamu skaitļošanas jaudu un resursus, padarot to mazāk pieejamu mazākām spēļu izstrādes komandām vai hobijiem.

Kopumā pētījums liecina, ka Deep Q Learning algoritmam ir potenciāls spēļu izstrādē, jo īpaši sarežģītām spēlēm, kurās nepieciešami adaptīvi un inteliģenti aģenti. Tomēr tā efektivitāte un iespējamība būtu jāizvērtē katrā gadījumā atsevišķi, ņemot vērā tādus faktorus kā izstrādes resursi, projekta prasības un mērķauditorija.

**3**

**2. Programmatūras prasību specifikācija – risinājuma mērķauditorijas izvēle un tās**

**raksturojums, programmatūras produkta un tā funkciju apraksts, programmatūras**

**produkta skice.**

Mērķauditorija:

Mērķauditorija: Mērķauditorija šai mākslīgā intelekta pārvaldītajai spēlei "Snake" ir ikviens, kas interesējas par spēļu spēlēšanu un vēlas izpētīt dziļas Q mācīšanās algoritmu pielietojumu spēļu izstrādē.

Risinājuma raksturojums:

Programmatūras risinājums būs mākslīgā intelekta vadīta spēle "Snake", kurā tiks izmantoti Deep Q Learning algoritmi, lai apmācītu aģentu spēlēt spēli. Spēlei būs grafisks interfeiss, kurā tiks attēlots spēles dēlis, čūska un ēdiens. Spēle būs interaktīva, kurā spēlētājs varēs kontrolēt čūskas kustību, un aģents arī mācīsies no spēlētāja gājieniem, lai uzlabotu savu spēli. Spēlē būs atalgojuma sistēma, kas apbalvos aģentu par ēdiena ēšanu un sodīs par sitieniem pa sienām vai savu ķermeni. Spēlei būs mācību posms, kurā aģents mācīsies spēlēt spēli, un testēšanas posms, kurā aģents spēlēs spēli pats.

Programmatūras produkta apraksts un tā funkcijas:

Programmatūras produkts būs mākslīgā intelekta vadīta spēle "Snake", kurai būs šādas funkcijas:

Grafiskā saskarne: Spēlei būs grafiskais interfeiss, kas attēlos spēles laukumu, čūsku un ēdienu. Interfeiss būs lietotājam draudzīgs un viegli saprotams.

Interaktīva spēle: Spēlētājs varēs kontrolēt čūskas kustību, izmantojot bulttaustiņus. Lai uzlabotu savu spēli, aģents mācīsies arī no spēlētāja kustībām.

Dziļas Q mācīšanās algoritms: Spēle izmantos Deep Q Learning algoritmus, lai apmācītu aģentu spēlēt spēli. Aģents mācīsies no balvām, ko tas saņems par ēdiena ēšanu, un sodiem par triecieniem sienām vai savam ķermenim.

Apmācības fāze: Spēlei būs mācību fāze, kurā aģents mācīsies spēlēt spēli. Aģents mācīsies no savas un spēlētāja spēles.

Testēšanas fāze: Spēlē būs testēšanas posms, kurā aģents pats spēlēs spēli. Lai spēlētu spēli, aģents izmantos iegūtās zināšanas.

Atlīdzības sistēma: Spēlē būs atlīdzības sistēma, kas aģentu apbalvos par ēdiena ēšanu un sodīs par sitieniem pa sienām vai savu ķermeni.

Programmatūras produktam būs grafisks interfeiss, kas attēlos spēles laukumu, čūsku un ēdienu. Uz spēles dēļa būs rezultātu rādītājs, kas sekos līdzi spēlētāja rezultātam. Testēšanas posmā uz spēles dēļa būs redzams arī aģenta rezultāts. Spēlētājs varēs kontrolēt čūskas kustību, izmantojot bulttaustiņus, un aģents mācīsies no spēlētāja gājieniem, lai uzlabotu savu spēli. Lai apmācītu aģentu spēlēt spēli, spēlē tiks izmantoti dziļās Q mācīšanās algoritmi. Aģents mācīsies no balvām, ko tas saņems par ēdiena ēšanu, un tiks sodīts par triecieniem sienām vai savam ķermenim. Spēlei būs apmācības fāze, kurā aģents mācīsies spēlēt spēli, un testēšanas fāze, kurā aģents pats spēlēs spēli.

**4**

**3. Programmatūras izstrādes plāns**.

Šajā projektā tiek izmantota pastiprinājuma mācīšanās, lai iemācītu čūsku orientēties vidē un ēst barību. Lai izveidotu spēli, mēs izmantosim Pygame - atvērtā pirmkoda bibliotēku videospēļu radīšanai, kas ir draudzīga iesācējiem un daudzplatformāta. Kad spēle būs izveidota, mēs koncentrēsimies uz pastiprinātas mācīšanās piemērošanu tai.

Lai to izdarītu, mums ir jāizveido trīs moduļi: vide (spēle), modelis (pastiprinājuma modelis gājienu prognozēšanai) un aģents (starpnieks starp vidi un modeli).

Spēles algoritms ietver nejauši izvietotu čūsku un ēdienu uz tāfeles, pēc tam aprēķinot čūskas pozīciju, izmantojot 11 vērtības. Atkarībā no konkrētiem nosacījumiem dažas vērtības var iestatīt uz nulli, pretējā gadījumā tiek izmantots vienādojums. Šīs 11 vērtības aģents izmanto, lai noteiktu pašreizējo spēles stāvokli.

Kad pašreizējais stāvoklis ir noteikts, aģents to nodod modelim, kas pēc tam paredz nākamo veicamo gājienu. Pēc nākamā gājiena veikšanas aģents aprēķina atlīdzību atkarībā no tā, vai čūska ēda barību (+10), vai spēle beidzās (-10), vai nekas nenotika (0). Pēc tam Q vērtība tiek atjaunināta, un modelis tiek apmācīts.

Apmācīšanas laikā Q vērtība tiek nejauši inicializēta, un sistēma nejauši izvēlas darbības, lai izpētītu spēli. Apmācības gaitā sistēma, izvēloties darbības, vairāk paļaujas uz savu neironu tīklu. Aģents atjaunina savu Q vērtību saskaņā ar Bellmana vienādojumu un saglabā datus par katru gājienu.

Modeļa kvalitāte ir izšķiroša projekta panākumiem. Čūskas gājienu pareizība ir atkarīga no modeļa kvalitātes. Tāpēc modelis tiks apmācīts pa daļām, lai nodrošinātu tā efektivitāti, paskaidrojot, kā modelis tiks apmācīts:

Modelis tiks apmācīts divās daļās: pirmajā daļā neironu tīklu apmācīs, izmantojot atkārtojamo atmiņu. Atkārtošanas atmiņā tiek saglabāta aģenta iepriekšējā pieredze, tostarp sākotnējais stāvoklis, veiktā darbība, iegūtā atlīdzība un nākamais stāvoklis. Neironu tīkls tiks apmācīts, izmantojot šo iepriekšējo pieredzi, lai prognozētu labāko rīcību, kas jāveic konkrētā stāvoklī.

Otrā apmācības daļa ir Q vērtību atjaunināšana, izmantojot Bellmana vienādojumu. Bellmana vienādojums ir rekursīva formula, kas atjaunina paredzamo nākotnes atlīdzību katram stāvokļa un darbības pārim, pamatojoties uz nākamā stāvokļa un darbības pāra paredzamo atlīdzību. Q-vērtības atspoguļo katra stāvokļa un darbības pāra paredzamo atlīdzību.

Mācību procesa laikā aģents līdzsvaro izpēti un ekspluatāciju. Sākumā aģents nejauši izpētīs vidi, lai gūtu pieredzi. Mācību gaitā aģents vairāk paļausies uz neironu tīkla prognozēm, lai izmantotu iegūtās zināšanas par vidi.

**5**

Apmācības process turpināsies, līdz aģenta veiktspēja sasniegs apmierinošu līmeni vai līdz tiks sasniegts maksimālais iterāciju skaits. Šajā brīdī tiks uzskatīts, ka aģents ir iemācījies efektīvi spēlēt spēli, un projekts tiks uzskatīts par veiksmīgu.

Kopumā projekta pamatā ir pastiprinājuma mācīšanās, un čūska tiek apmācīta ēst barību savā vidē, izmantojot Pygame. Projektā ir jāizveido trīs moduļi: Vide, modelis un aģents. Modelis tiek apmācīts, izmantojot atkārtojuma atmiņu un Bellmana vienādojumu, lai paredzētu labākās darbības, kas jāveic noteiktā stāvoklī. Apmācīšanas process līdzsvaro izpēti un ekspluatāciju un turpinās, līdz aģents sasniedz apmierinošu veiktspējas līmeni vai tiek sasniegts maksimālais iterāciju skaits.

**4. Lietotāja ceļvedis.**

Lai palaistu šo spēli, vispirms izveidojiet vidi anaconda vai (jebkurā platformā). Pēc tam instalējiet nepieciešamos moduļus, piemēram, Pytorch (DQ mācību modelim), Pygame (spēles vizualizācijai) un citus pamata moduļus.

Pēc tam palaidiet agent.py failu tikko izveidotajā vidē, un tad sāksies apmācība, un jūs redzēsiet divus grafiskos interfeisus, no kuriem viens attēlo apmācības gaitu, bet otrs - mākslīgā intelekta vadīto čūsku spēli.

Pēc noteikta rezultāta sasniegšanas jūs varat beigt spēli, un tikko apmācītais modelis tiks saglabāts ceļā, kas definēts models.py saglabāšanas funkcijā.

Nākotnē šo apmācīto modeli var izmantot, vienkārši mainot kodu failā agent.py, kā parādīts turpmāk:

Kā instalēt un palaist:

1. Instalējiet Anaconda izplatīšanu no https://www.anaconda.com/products/distribution
2. Palaidiet Anaconda.
3. Palaidiet Anaconda CMD.exe Promt.
4. Ierakstiet CMD:

* conda create -n pygame\_env python=3.7
* conda activate pygame\_env
* pip install pygame
* Nokopējiet un ielīmējiet PyTorch instalēšanas komandu no https://pytorch.org/get-started/locally/ (piemērs: pip install torch torchvision).
* Instalējiet matplotlib un ipython, izpildot šādu komandu: pip install matplotlib ipython
* Palaidiet CD uz projekta direktoriju, izpildot komandu (piemērs: CD C:\Users\User\Desktop\SnakeAI).
* Palaidiet spēli, ievadot šādu komandu: python agent.py

Sāksies apmācība, un tiks parādītas divas grafiskās saskarnes: vienā tiks parādīta apmācība, bet otrā - mākslīgā intelekta kontrolēta čūskas spēle.

Pēc noteikta rezultāta sasniegšanas var beigt spēli, un jaunais apmācītais modelis tiks saglabāts ceļā, kas definēts models.py saglabāšanas funkcijā.

Nākotnē šo apmācīto modeli var izmantot, vienkārši mainot kodu failā agent.py.

**6**

**5. Piemērotās licences pamatojums.**

Programmatūras projekta licences izvēle ir atkarīga no dažādiem faktoriem, piemēram, projekta veida, paredzētā lietojuma un izstrādātāju vēlmēm.

Šim projektam būtu piemērota tāda populāra un liberāla licence kā MIT licence. Šī licence ļauj ikvienam bez ierobežojumiem izmantot, modificēt un izplatīt programmatūru gan komerciāliem, gan nekomerciāliem mērķiem. Tā ietver arī ierobežotas atbildības un garantijas noteikumus.

MIT licence ir plaši izmantota atvērtā koda kopienā un ir saderīga ar citām licencēm, tādējādi atvieglojot šī projekta integrāciju ar citiem projektiem. Turklāt tā ļauj brīvi izmantot un modificēt programmatūru, vienlaikus nodrošinot zināmu juridisko aizsardzību izstrādātājiem.

**7**