Neuroewolucja

Dokumentacja wstępna

Kacper Kula
Krzysztof Pierczyk

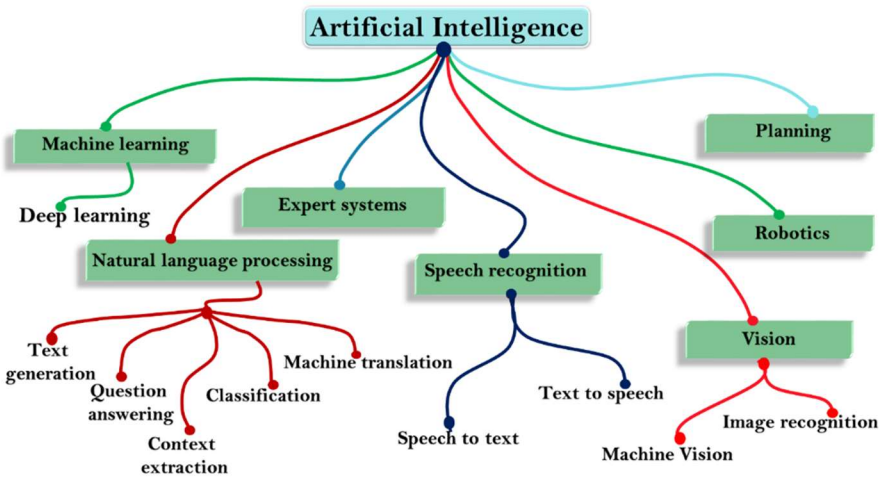
12 kwietnia 2020
Warszawa

WSTĘP

W drugiej połowie dwudziestego wieku rozpoczęła się, trwająca po dziś dzień, rewolucja technologiczna. Szybki rozwój elektroniki oraz, napędzany przez niego, rozwój informatyki, który miał być jedynie przedłużeniem rewolucji przemysłowej, ujawnił ludzkości wagę nowej wartości, nieznanej w poprzednich epokach – informacji. Komputery napędzane przez coraz to szybsze jednostki obliczeniowe dostarczają nam każdego dnia niezmierzoną ilość informacji, która w dzisiejszych czasach zaczyna jawić się jako najbardziej wartościowe dobro. Mimo iż wzrost szybkości współczesnych komputerów nie zwalnia, wciąż jest on ograniczony. Pociąga to za sobą potrzebę intensywnego rozwoju teoretycznych aspektów informatyki celem lepszej utylizacji dostępnej mocy obliczeniowej.

Sztuczna inteligencja

Klasyczne algorytmy, rozumiane jako ścisłe schematy postępowań mające gwarantować osiągnięcie określonego celu, były wystarczające przez kilka dekad. Dały one solidne podstawy do dalszego rozwoju technologicznego, lecz w miarę wzrostu możliwości związanych z pozyskiwaniem i agregacją informacji, stały się one niewystarczające. Wymusiło to na inżynierach stworzenie nowego podejścia do problemu i poskutkowało narodzeniem się dziedziny informatyki nazywanej dzisiaj powszechnie **sztuczną inteligencją**. Termin ten jest na tyle obszerny, że ciężko w tym momencie podać jego precyzyjną definicję, która pozwoliłaby na sztywną klasyfikację metod obliczeniowych jako mogących i niemogących nosić miana AI (*ang. Artificial Intelligence*). Jednak na przestrzeni lat wyłonił się pewien podział wewnątrz samego terminu, który wprowadza odrobinę porządku do tematu.



Rysunek 1. Działy sztucznej inteligencji

Według tego podziału, algorytmy sztucznej inteligencji możemy skategoryzować na te, zajmujące się **optymalizacją** (lub innymi słowy przeszukiwaniem przestrzeni) i **uczenie maszynowe**. Przeznaczenie pierwszej z tych grup wydaje się być oczywiste. Druga z nich zajmuje się z kolei szeroko pojętym przetwarzaniem danych i charakteryzuje się umiejętnością swego rodzaju „uczenia się", zarówno na bazie samych danych jak i własnych doświadczeń.

Rysunek 2. Porównanie klasycznego podejścia programistycznego z uczeniem maszynowym

W odróżnieniu do klasycznego podejścia programistycznego, algorytmy oparte o uczenie maszynowe mają za zadanie nie tyle znajdowanie samych odpowiedzi, na postawione pytania, ile wytworzenie, na bazie otrzymanych danych, pewnych reguł, które pozwolą im funkcjonować w środowisku. Jedną z najpopularniejszych, w dzisiejszych czasach, metod uczenia maszynowego są **sztuczne sieci neuronowe** (*ang. Artificial Neural Networks, ANN*).

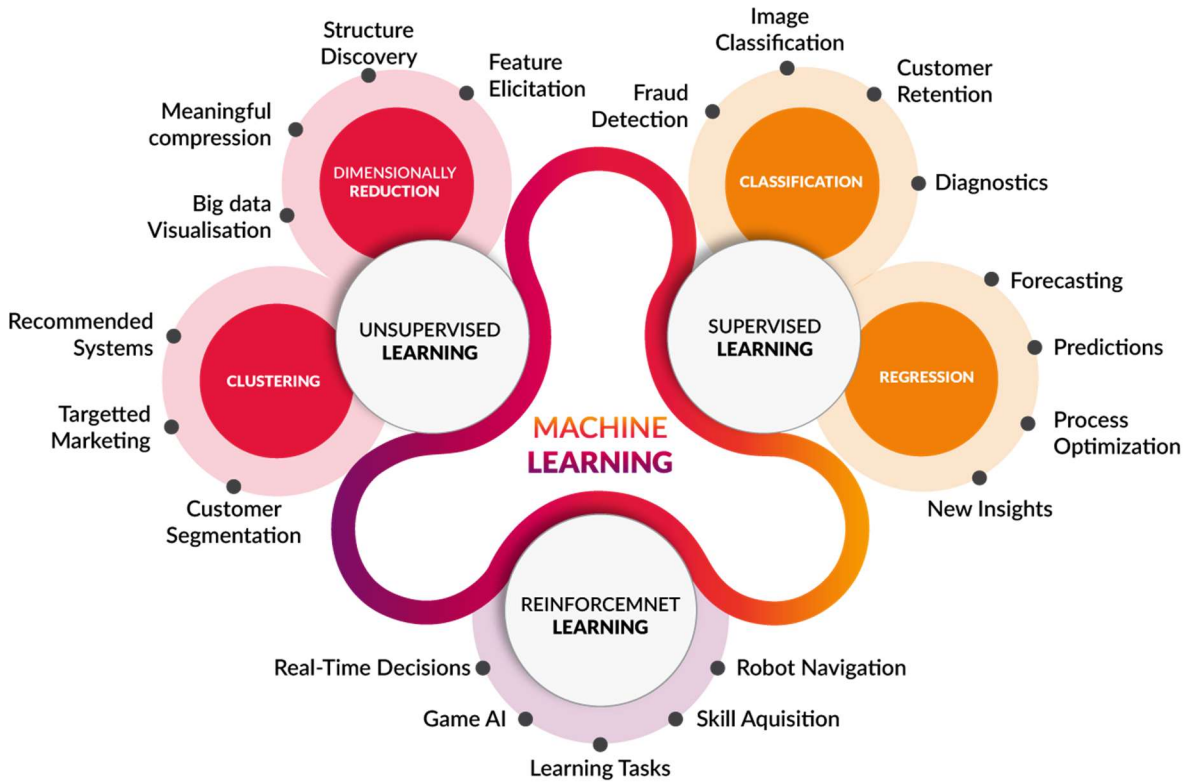
Tematyka projektu

W ramach realizowanego projektu postanowiliśmy zająć się tematyką uczenia sieci neuronowych. Najpopularniejsze metody w tej dziedzinie opierają się na idei algorytmu **spadku wzdłuż gradientu** (*ang. gradient descent method*), zaproponowanej oryginalnie przez Augustina Louisa Cauchy'ego w 1847 roku. Jest to algorytm poszukiwania optimum lokalnego funkcji celu. Idea modelowania pewnych systemów nieliniowych za pomocą

sztucznych sieci neuronowych narodziła się już w połowie dwudziestego wieku, lecz brak skutecznych metod ich uczenia sprawił, że nie zdobyły one wielkiej popularności przez kolejnych kilkadziesiąt lat. Dopiero odkrycie, w jaki sposób zastosować metodę spadku wzdłuż gradientu (szczególnie w jej stochastycznym wariancie), dzięki wstecznej propagacji błędu popełnianego przez sieć, do optymalizowania jej parametrów, połączone ze wzrostem dostępnej mocy obliczeniowej ukazały potencjał w nich drzemiący.

Metoda ta ma jednak szereg wad, wśród których jedną z najbardziej znaczących jest brak umiejętności opuszczenia lokalnego minimum. Poszukiwanie metod uczenia ANN, które byłyby pod tym względem bardziej odporne, doprowadziło do powstania nowej dziedziny uczenia maszynowego zwanej **neuroewolucją**. Zakłada ona wykorzystanie algorytmów ewolucyjnych do optymalizacji parametrów sieci. Mimo, że są to algorytmu heurystyczne, czyli takie, dla których nie ma gwarancji znalezienia optimum funkcji w skończonym czasie, praktyka pokazuje, że potrafią one sprawdzać się w przypadku niektórych scenariuszy o wiele lepiej niż metody klasyczne oraz posiadają względem nich dodatkowe atuty np. w postaci możliwości dostrajania nie tylko wag sieci, ale również jej struktury. To właśnie tą odnogą uczenia ANN zajmiemy się w ramach projektu.

NEUROEWOLUCJA



Rysunek 3. Działy uczenia maszynowego

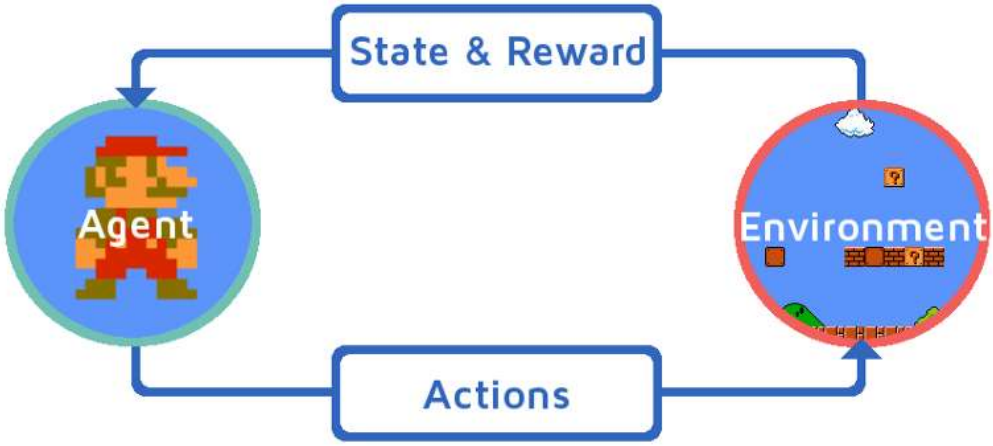
Uczenie maszynowe dzieli się zwykle na trzy grupy. Są to kolejno **uczenie nadzorowane** (*ang. Supervised Learning*), **uczenie nienadzorowane** (*ang. Unsupervised Learning*) oraz **uczenie ze wzmocnieniem** (*ang. Reinforcement Learning*).

Uczenie nadzorowane polega na podaniu algorytmowi pewnego zbioru danych uczących, który zawiera dane wejściowe oraz pożądane odpowiedzi, związane z tymi danymi oraz dostosowanie parametrów algorytmu tak, aby z jednej strony jego odpowiedzi pokrywały się z odpowiedziami ze zbioru uczącego, a z drugiej, aby potrafił on uogólnić zdobytą „wiedzę" na przypadki, które zostaną mu dostarczone w przyszłości. Przykładem uczenia nadzorowanego jest uczenie ANN z wykorzystaniem metody (stochastycznego) spadku wzdłuż gradientu.

Uczenie nienadzorowane służy głównie do znajdowania wśród danych wejściowych algorytmu pewnych wzorców i zależności, które nie byłyby możliwe do spostrzeżenia w przypadku analizowania ich przez człowieka. Do algorytmu są dostarczane jedynie dane

wejściowe bez podawania oczekiwanych wyjść algorytmu. Do algorytmów z tej dziedziny możemy zaliczyć m.in. algorytm K najbliższych sąsiadów.

Uczenie ze wzmocnieniem jest to dziedzina, która w pewnym stopniu próbuje połączyć najlepsze cechy dwóch poprzednich metod. Zakłada ona, że algorytm „uczy się" jedynie na bazie własnych interakcji ze środowiskiem, a jedyną informacją, którą otrzymuje jest ocena podejmowanych przez niego decyzji.



Rysunek 4. Schemat uczenia ze wzmocnieniem

Uczenie maszynowe, a neuroewolucja

Uczenie sztucznych sieci neuronowych za pomocą algorytmów ewolucyjnych może być stosowane zarówno w dziedzinie uczenia nadzorowanego, jak i uczenia ze wzmocnieniem. Polega ono na zakodowaniu informacji o szkolonej sieci (sieciach) w postaci obiektów będących osobnikami pewnej populacji. W każdej iteracji algorytmu ewolucyjnego osobniki są poddawane ocenie, a następnie, na bazie tej oceny, wybierane są te, które przekażą geny następnej generacji. Osobniki dobierane są w pary, które są następnie krzyżowane, wytwarzając tym samym nowe osobniki. Aby zmniejszyć presję selekcyjną algorytmu i zapobiec zbyt wczesnemu zbiegnięciu populacji do optimum lokalnego, pojedyncze osobniki poddawane są losowym mutacjom, które mają zapewnić napływ nowych genów w populacji, a tym samym zwiększyć eksplorację przestrzeni parametrów sieci.

Nomenklatura

Nomenklatura funkcjonująca w dziedzinie neuroewolucji jest taka sama, jak w przypadku tradycyjnych algorytmów ewolucyjnych. W ramach sprowadzenia rozważań na bardziej precyzyjne tory, warto przypomnieć kilka terminów i podać ich interpretacje.

1. **Populacja** – zbiór osobników, które reprezentują uczoną sieć (sieci) neuronową
2. **Generacja** – aktualna populacja w danej iteracji algorytmu
3. **Osobnik** – obiekt reprezentujące sieć neuronową lub pewien jej element. Na gruncie neuroewolucji osobnikiem takim może być cała sieć neuronowa, warstwa sieci, pojedynczy neuron lub połączenie między neuronami. Najpopularniejszym podejściem jest reprezentowanie całych sieci w postaci pojedynczego osobnika.
4. **Fenotyp** – zbiór cech jednoznacznie reprezentujących sztuczną sieć neuronową
5. **Genotyp** – zbiór zakodowanych cech sieci neuronowej (zakodowany fenotyp) lub zbiór danych pozwalających wygenerować fenotyp, w postaci umożliwiającej dogodne przeprowadzanie operacji krzyżowania i mutacji.
6. **Gen** – pojedynczy parametr genotypu
7. **Chromosom** – zbiór genów kodujących pojedynczą cechę

Powyższa nomenklatura pochodzi oryginalnie z dziedziny algorytmów genetycznych, lecz rozpowszechniła się także wśród innych algorytmów ewolucyjnych.

Problemy stawiane przez neuroewolucję

Przy projektowaniu algorytmu neuroewolucyjnego możemy wyróżnić dwie zasadnicze trudności. Pierwsza z nich, zaprezentowana poniekąd w poprzednim akapicie, to **dobór odpowiedniej reprezentacji sieci za pomocą genotypu**. Istnieje bardzo wiele sposobów, na które można to zrobić i właśnie na bazie nowych pomysłów dotykających tego zagadnienia powstają najbardziej efektywne algorytmy neuroewolucyjne. Projektując genotyp należy wziąć pod uwagę m.in. sposób w jaki będą przeprowadzane krzyżowanie i mutacja, złożoność obliczeniową i pamięciową samych operacji, jakie parametry sieci będą podlegały modyfikacji w kolejnych generacjach algorytmu. To właśnie projekt genotypu określa, czy algorytm będzie w stanie wpływać tylko na wagi sieci, czy także dostosowywać jej topologię.

Drugą, chociaż być może nie tak złożoną kwestią jest **sposób oceny poszczególnych osobników** w populacji. Czy oceniać każdą z ich decyzji z osobna, czy może cały zbiór decyzji na pewnym horyzoncie oceny? Jakie wartości powinny przybierać oceny? Ciągłe, dyskretne, a może symboliczne? Zagadnienie to nie jest charakterystyczne dla neuroewolucji. Występuje ono w przypadku wszystkich metod uczenia maszynowego ze wzmocnieniem. Rozwiązanie tego problemu jest zależne od samej natury problemu i podstawowe pomysły na implementację systemu oceniania przychodzą często już na etapie określania problematyki. W tym miejscu warto zaznaczyć, że neuroewolucja kwalifikować się może także do metod uczenia nadzorowanego np. w przypadkach, kiedy oceną osobnika, reprezentującego pojedynczą sieć neuronową, jest sumaryczny błąd popełniany na zbiorze uczącym.

Klasyfikacja algorytmów neuroewolucyjnych

Algorytmy ewolucyjne można podzielić na dwa sposoby, z których jeden bierze pod uwagę zakres dostrajanych parametrów sieci, a drugi sposób kodowania fenotypu

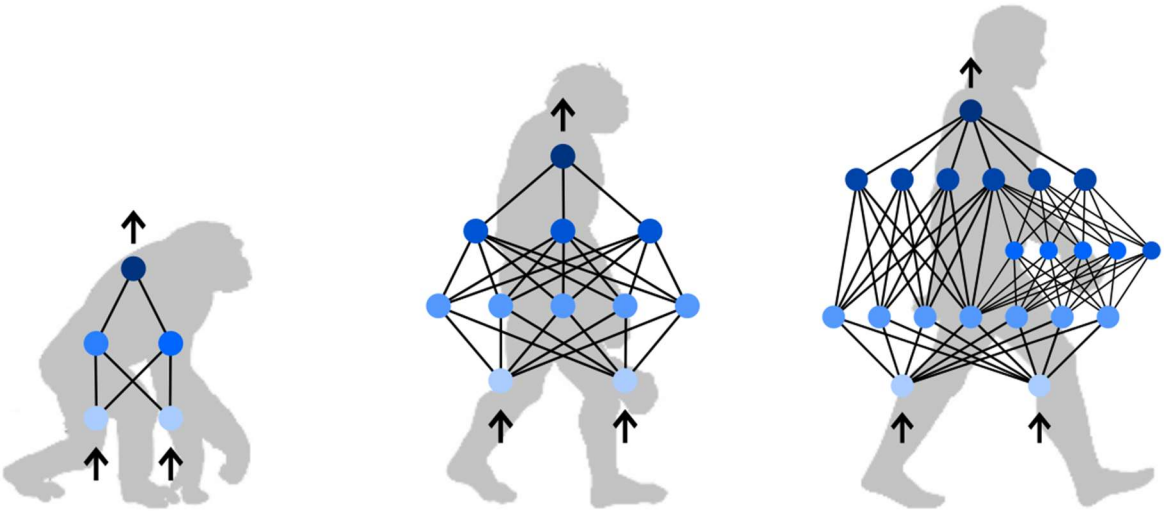
I. Algorytmy **modyfikujące wagi sieci** / Algorytmy **modyfikujące topologię i wagi sieci** – dzieląc algorytmy modyfikujące topologię dalej, można wyszczególnić te, które modyfikują wagi równolegle z topologią i te, które modyfikują obie grupy parametrów (możliwie) niezależnie
II. Algorytmy o **kodowaniu bezpośrednim** / Algorytmy o **kodowaniu pośrednim** – jeżeli genotyp da się jednoznacznie odwzorować na zbiorze cech sieci neuronowej, mamy do czynienia z algorytmem o kodowaniu bezpośrednim. Jeżeli chromosomy genotypu podają jedynie wartości służące jako baza do wygenerowania jednej z możliwych sieci, mówimy o kodowaniu pośrednim. Do algorytmów o kodowaniu bezpośrednim należą m.in.:
    a. Ewolucja neurogenetyczna (*ang. Neuro-genetic evolution*) [E. Ronald, 1994]
    b. Neuroewolucja rosnących topologii (*ang. NeuroEvolution of Augmenting Topologies, NEAT*) [Stanley and Miikkulainen, 2002]

    Z kodowania pośredniego korzystają natomiast:

    a. Kodowanie komórkowe (*ang. Cellular Encoding*) [F. Gruau, 1994]
    b. Neuroewolucja rosnących topologii oparta o hipersześciany (*ang. Hypercube-based NeuroEvolution of Augmenting Topologies, HyperNEAT*) [Stanley, D'Ambrosio, Gauci, 2008]

Neuroewolucja a podejście klasyczne

Jak wspomniano już wcześniej, klasyczne podejście do uczenia sieci neuronowych oparte na wstecznej propagacji błędu, mimo, że w efektywny sposób pozwala stroić parametry rozległych sieci neuronowych, składających się z milionów połączeń między neuronami, ma wiele wad. Jedną z najpoważniejszych wydaje się **brak wbudowanych mechanizmów pozwalających na wydostanie się algorytmu z optimum lokalnego**. Poza tym sieci uczone tymi metodami mają tendencję do **przeuczania się**. Aby poprawić taki stan rzeczy, na algorytmy klasyczne nakłada się różne modyfikacje, jak wczesne zatrzymanie uczenia, czy okresowa dezaktywacja niektórych neuronów w sieci.



Rysunek 5. Poglądowe przedstawienie neuroewolucji

Neuroewolucja podchodzi do problemu z zupełnie innej strony i proponuje zastosowanie metod, które z natury oferują dużą **kontrolę nad tendencjami algorytmu do eksploracji przestrzeni i eksploatacji dotychczasowych rozwiązań**. Ponadto umożliwia ona **silne zrównoleglenie obliczeń** ze względu na fakt, że utrzymuje ona populację o stałej liczebności przez wszystkie iteracje algorytmu.

Ponadto algorytmy neuroewolucyjne umożliwiają **dostosowywanie topologii sieci do konkretnego problemu**. W przypadku podejścia klasycznego, wybór topologii, dokonywany a priori, jest jedną z najbardziej kłopotliwych kwestii. Problemy o różnej naturze są lepiej modelowane przez niektóre struktury neuronowe aniżeli przez inne. Podejście oparte o algorytmy ewolucyjne zwalnia projektanta z obowiązku wyboru topologii i pozwala na jej dynamiczne dostosowywanie wraz ze zmieniającymi się w otoczeniu warunkami.

ZAKRES PROJEKTU

Cel projektu

Celem naszego projektu jest poznanie, przetestowanie oraz porównanie wybranych technik neuroewolucyjnych w środowisku OpenAI-gym. Początkowym obiektem testowym będzie gra Pong z konsoli Atari 2600, jednak ustandaryzowanie środowiska umożliwia nam także w miarę bezproblemowe wykorzystanie innej gry.

Podstawowymi założeniami projektu są:

- Wykorzystanie genetycznego algorytmu NEAT do stworzenia sieci neuronowej zdolnej do efektywnej gry w Pong
- Wykorzystanie innego algorytmu genetycznego z kategorii modyfikujących jedynie wagi sieci w tym samym celu
- Porównanie wyników

Dodatkowym celem jest porównanie wspomnianych algorytmów z klasycznym podejściem wykorzystującym technikę wstecznej propagacji błędu, jednak z uwagi na fakt, że jest ona nieprzystosowana do metodyki oferowanej przez OpenAI-gym (brak zbioru uczącego), może się to okazać niewykonalne w dysponowanym przez nas czasie.

Narzędzia

OpenAI-gym

OpenAI-gym to interfejs pomiędzy środowiskiem testowym a obiektem podejmującym decyzję, „graczem” lub „agentem”. Interfejs ten dostarcza możliwe do podjęcia decyzje; po wykonaniu kroku w symulacji zwracana jest informacja o stanie. Do informacji tych należą m.in. obserwacje, na podstawie których gracz może dokonać następnej decyzji oraz nagroda, informująca o jego efektywności.

Gra Pong

Gra Pong w wersji na Atari 2600, 8-bitową konsolę z 1977, jest doskonałym kandydatem na obiekt testowy z uwagi na swoje minimalne wymagania. Łączna ilość RAM wykorzystywana przez nią to raptem 128 bajtów. Na jej oprawę graficzną składa się statyczna plansza, na której jedynymi ruchomymi obiektami są liczniki punktów, platformy sterowane przez graczy oraz piłka. Grę możemy obserwować, korzystając z dwóch różnych zbiorów danych: obrazu lub pamięci RAM. Pierwszy z tych zbiorów wartości to tablica 210x160 pikseli z trzema

kanałami kolorów. Drugi to wspomniane 128 bajtów. Z uwagi na znacznie mniejszą ilość danych wejściowych, początkowo skupimy się na wersji z obserwacją RAM-u.

NEAT (NeuroEvolution of Augmenting Topologies)

Algorytm genetyczny z kategorii modyfikujących wagi oraz topologię sieci neuronowych. Występujące w nim osobniki to całe sieci neuronowe. NEAT przedstawia propozycje czterech mechanizmów, które pozwalają w efektywny sposób ewoluować osobnikom. Są to:

- Kodowanie
- Krzyżowanie
- Mutacja
- Powstawanie gatunków (speciation)

Kodowanie

Jednym z problemów sieci neuronowych jest problem konkurujących konwencji (*ang. competing conventions problem*). Objawia się on w sytuacjach, kiedy sieci, pomimo różnych kodowań, przedstawiają tę samą strukturę. W takich sytuacjach może dojść do utraty informacji. W celu uniknięcia takich możliwości NEAT wykorzystuje historyczne pochodzenie genów. Dodatkowo sposób kodowania jest podłożem dla następnych zagadnień.

Krzyżowanie

Krzyżowanie to standardowa metoda tworzenia nowych populacji osobników znana z algorytmów genetycznych. NEAT wykorzystując wspomniane kodowanie z uwzględnieniem informacji historycznej potrafi rozpoznać, które połączenia w sieci neuronowej są tożsame. W sytuacji, gdy rodzice mają tę samą parę połączeń, potomek dziedziczy losowy gen z pary. Jeśli gen występuje u jednego z rodziców, to dziedziczone są geny od lepszego rodzica.

Mutacja

Aby dodać losowej różnorodności osobnikom, w algorytmach ewolucyjnych stosuje się także mutacje. Mutacje w NEAT przyczyniają się do zwiększenia złożoności osobnika. Są to: dodanie połączenia oraz dodanie neuronu. To dzięki tym mutacjom sieci mają szanse na stworzenie znaczącej struktury. Wykorzystując specjalne kodowanie genotypu dba się, by mutacje te były w sensowny sposób przenoszone pomiędzy osobnikami oraz by osobniki były kompatybilne. Oprócz tego dostępne są standardowe mutacje wartości wag.

Powstawanie gatunków

Ostatnim z podstawowych problemów wszelakich algorytmów optymalizujących jest utykanie w optimach lokalnych. Działaniem zapobiegającym temu zjawisku wykorzystywanym w NEAT jest dzielenie osobników na gatunki w zależności od ich podobieństwa. Podczas wybierania następnej populacji, osobniki są gatunkowane,

a następnie wybierane z uwzględnieniem tego podziału, by zachować jak najefektywniejszą różnorodność gatunkową. Pozwala to na znacznie zwiększenie przestrzeni poszukiwań, co z kolei przekłada się na większe prawdopodobieństwo znalezienia się w sąsiedztwie coraz to lepszych optimum.

Pozostałe

Wzorem OpenAI-gym, projekt zostanie napisany w języku Python. Planujemy wykorzystać takie biblioteki jak NEAT-Python, TensorFlow oraz DEAP.

BIBLIOGRAFIA

[1] https://towardsdatascience.com/gradient-descent-vs-neuroevolution-f907dace010f

[2] https://towardsdatascience.com/a-primer-on-the-fundamental-concepts-of-neuroevolution-9068f532f7f7

[3] https://www.freecodecamp.org/news/a-brief-introduction-to-reinforcement-learning-7799af5840db/

[4] https://www.nature.com/articles/s42256-018-0006-z

[5] https://towardsdatascience.com/coding-deep-learning-for-beginners-types-of-machine-learning-b9e651e1ed9d

[6] http://www.miroslawmamczur.pl/czym-jest-uczenie-maszynowe-i-jakie-sa-rodzaje/

[7] https://www.javatpoint.com/subsets-of-ai

[8] Wspomaganie sterowania statkiem za pomocą ewolucyjnych sieci neuronowych, Mirosław Łącki, Akademia Morska w Szczecinie, 2008

[9] Neuroewolucja topologii sieci neuronowych z wykorzystaniem algorytmów genetycznych, Piotr Boryczko, Politechnika Krakowska, 2011

[10] http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf