Sztuczne Sieci Neuronowe
Sprawozdanie z projektu

Łukasz Rados, Wojciech Kusa
Wydział Fizyki i Informatyki Stosowanej
Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

14 września 2015

1 Wprowadzenie

Celem opisywanego projektu było zaimplementowanie i próba nauczenia sieci neuronowej grania w prostą grę komputerową. Genezą projektu jest publikacja [1], w której autorzy próbują nauczyć sieć neuronową za pomocą metody tzw. *deep learning* algorytmów grania w różne gry komputerowe wydane na platformę Atari. Naszą motywacją było stworzenie gry komputerowej wzorującej się na grze *T-Rex Runner* znajdującej się w przeglądarce Google Chrome [2], a następnie zaimplementowanie takiej struktury sieci neuronowej, która byłaby w stanie nauczyć się skutecznie grać w tę grę.

Rozważane zostały różne rodzaje reprezentacji planszy gry (opisujące różne poziomy abstrakcji) oraz wpływ technik uczenia sieci na jakość działania. Ze względu na ograniczenia implementacyjne przetestowane zostało działanie tylko sieci o strukturze radialnej. Uzyskane wyniki (pomimo iż opracowane przypadki były relatywnie proste) są zadowalające oraz stanowią podstawę i dobry punkt wyjścia do dalszego rozwoju projektu.

Projekt został wykonany w języku Python 2.7/3.4 z wykorzystaniem bibliotek do obliczeń numerycznych NumPy, SciPy oraz biblioteki wspomagającej proces tworzenia gier komputerowych PyGame.

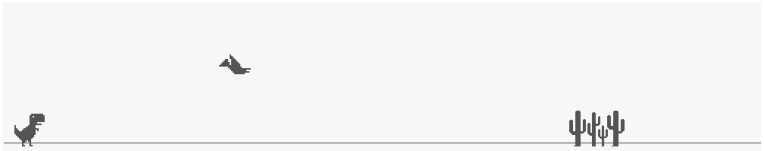
2 Podstawy teoretyczne

Problem przyswajania przez algorytmy uczenia maszynowego algorytmów dotyczących innych dziedzin (zadań) jest najczęściej sprowadzany do problemu klasyfikacji i rozpoznawania wzorca. Podstawowymi elementami, które są różne dla tych zadań jest sposób reprezentacji przestrzeni stanu (problemu) oraz sprzężenie zwrotne sygnalizujące jakość odpowiedzi układu.

2.1 Gra

W ramach tego projektu zaimplementowana została gra komputerowa bazująca na grze *T-Rex Runner*. *T-Rex Runner* jest jednowymiarową grą operująca na niewielkiej przestrzeni możliwych stanów (kilka rodzajów kaktusów oraz latający pterodaktyl) oraz dwóch możliwych ruchach (skok i przykucnięcie). Dzięki tym właściwościom i nieskomplikowanemu interfejsowi człowiekowi bardzo prosto jest opisać jej reguły a skuteczność grania sprowadza się tylko do refleksu.

Gra została zaimplementowana z wykorzystaniem biblioteki PyGame do tworzenia gier komputerowych [3]. Zrzut ekranu przedstawiający działającą grę znajduje się na Rysunku 1.Wychodząc od najprostszej wersji gry czyli z dyskretnym przesunięciem oraz trzema rodzajami obiektów do testów działania sieci neuronowej przygotowano kilka dodatkowych rozszerzeń utrudniających rozgrywkę takich jak ciągły skok dinozaura czy dodatkowy typ kaktusa.



Rysunek 1: Zrzut ekranu przedstawiający grę.

2.2 Reprezentacja problemu

W wersji podstawowej gry plansza opisana jest w postaci macierzy liczb całkowitych o rozmiarze 10×2 będącej reprezentacją obiektów znajdujących się na planszy. Przykładowo, plansza przedstawiona na Rysunku 1 opisana będzie przez macierz:

$$\begin{bmatrix} 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \end{bmatrix}$$

Reprezentacja ta, wydaje się prosta ale generuje problemy przy próbie uczenia takiej sieci z powodu dużej liczby połączeń w warstwie ukrytej. Natomiast nauczona sieć dawała bardzo dobre rezultaty, nawet gdy do położenia obiektów wprowadzono rozmyte wartości (prawdopodobieństwo wystąpienia obiektu w tym miejscu).

Rozważania była też niskopoziomowa odmiana powyższej metody gdzie plansza byłą reprezentowana przez piksele, a nie obiekty. Jednak opisanie planszy za pomocą pikseli (nawet binarnie) okazało się zadaniem znacznie wykraczającym poza możliwości normalnych komputerów, gdyż trenowanie takiej sieci zajmowało zdecydowanie za dużo czasu a sieć pomimo swoich ogromnych rozmiarów (nawet przy niewielkiej rozdzielczości ekranu

było to co najmjiej 10^7 połączeń) nie dawała sobie rady z rozwiązaniem problemu. Podstawową kwestią jest tutaj konieczność dokonywania meta-klasyfikacji (piksele trzeba najpierw pogrupować w obiekty i dopiero wtedy można dokonywać klasyfikacji ruchów dinozaura) przez sieć do czego potrzebna jest inna struktura i inny sposób uczenia sieci.

Z powyższych względów zdecydowano się też na przetestowanie innego sposobu reprezentacji planszy gry. Plansza opisana została przez n pierwszych elementów znajdujących się na niej. Pominięte zostały wszystkie puste pola co znacznie zmniejszyło liczbę danych wejściowych ale jednocześnie nie uprościło znacząco rozgrywki gdyż w dalszym ciągu gracz musi zadecydować czy ta względna odległość jest wystarczająca aby wykonać skok i ominąć przeszkodę.

Na początku opracowana została reprezentacja, w której każdy obiekt opisany jest przez 3 wielkości (x_1 - początkowe położenie poziome, x_2 - końcowe położenie poziome oraz y - typ obiektu, Rysunek 2). Tak stworzoną reprezentację z łatwością można rozbudowywać o dodatkowe parametry "utrudniające" rozgrywkę takie jak:

- liczbę analizowanych obiektów,
- rozmycie położenia obiektów,
- wprowadzenie rozmiaru obiektu w osi y zamiast jego typu.

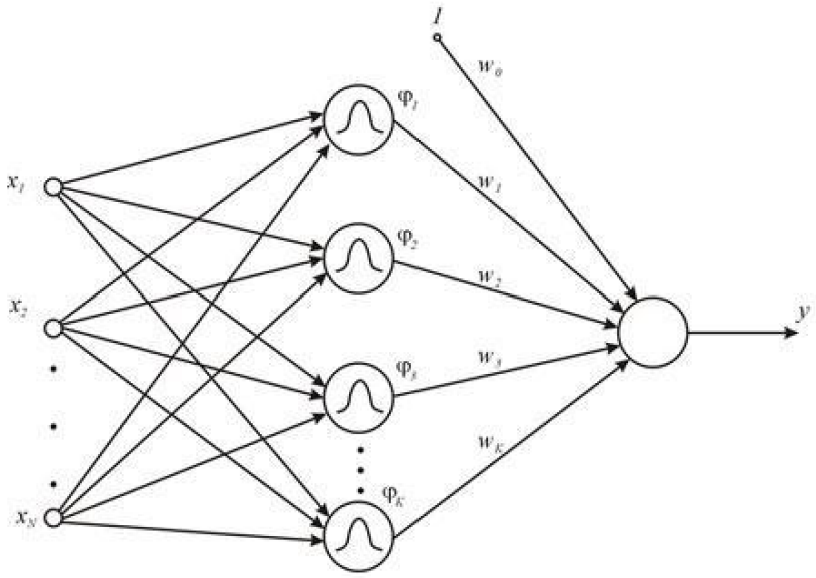
Rysunek 2: Drugi sposób reprezentacji planszy gry.

2.3 Sieć Neuronowa

Sieć radialna (ang. *radial basis function network, RBFN*) jest odmianą sigmoidalnej iteracyjnej sieci neuronowej. Neuron takiej sieci reprezentuje hipersferę dokonując podziału kołowego. Jego funkcją aktywacji jest funkcja radialna, na przykład $\phi(r) = e^{-\frac{r^2}{\beta}}$. Model sieci radialnej przedstawiony został na Rysunku 3.

Proste sieci radialne działają na zasadzie interpolacji wielowymiarowej, w której n różnych wektorów wejściowych jest odwzorowywanych z przestrzeni N-wymiarowej w zbiór k liczb rzeczywistych $d_i, i = 1, 2, ..., k$.

Sieci radialne z powodzeniem stosowane były w rozwiązywaniu problemów klasyfikacyjnych, zadaniach aproksymacyjnych oraz zagadnieniach predykcji czyli wydają się idealnym narzędziem do nauki algorytmu gry w opisanych wyżej sposobach reprezentacji



Rysunek 3: Model sieci radialnej.

przestrzeni stanów.

W projekcie wykorzystano implementację sieci radialnej pyradbas [4] dostosowując ją do specyfiki problemu (generowanie odpowiedzi na bieżąco w trakcie gry).

3 Wyniki

Ze względu iż trenowanie sieci podczas reprezentacji obejmującej macierz obiektów znajdujących się na planszy zajmowało bardzo dużo czasu sposób ten został uznany za nieefektowny przy wybranej strukturze sieci i pominięto go podczas dalszych eksperymentów.

Przestestowane zostało zatem działanie sieci neuronowej dla reprezentacji względnego położenia obiektów na planszy. Sieć trenowana jest za pomocą obserwacji ruchów gracza czlowieka przy konkretnym stanie planszy podczas kilku uruchomień gry, a następnie testowana jest gdy podczas uruchomionej gry musi "w locie" przewidywać następny ruch.

Średnie wartości punktów uzyskane przez sieć w zależności od liczby analizowanych obiektów przedstawione zostały w Tablicy 1. Gdy sieć bierze pod uwagę tylko jeden obiekt (na wejściu otrzymuje 3 parametry) to potrafi bezbłędnie przewidzieć następny ruch. Przy zwiększaniu liczby parametrów jej sprawność pogarasza się znacząco - przy 4 obiektach nie potrafi wykonać nawet jednego skoku. Wyniki te nie są uzależnione od

Tablica 1: Wyniki uzyskane przez sieć neuronową.

Liczba analizowanych obiektów	**Średni wynik sieci**	**Odchylenie standardowe**
1	inf	0
2	635.2	21.1
3	105.8	16.4
4	72.5	3.3

rozmiaru zbioru treningowego - dla 50 elementów treningowych sieć uzyskuje takie same wyniki jak na dla zbioru treningowego zawierającego 2000 przypadków.

Wartości pobrane z planszy były dodatkowo zaburzane niewielkimi wartościami szumem gaussowskim

Sprawdzone zostało też, jak zmienia się wartość wyniku uzyskanego przez sieć neuronową w zależności od rozmycia położenia obiektów na planszy (przyjęto że 1- całkowite rozmycie, 0 - brak rozmycia). Wyniki symulacji znajdują się na Rysunku 4. Gdy rozmycie nie występuje sieć potrafi grać w nieskończoność, co można zaobserwować na wykładniczym wzroście liczby zdobytych punktów przez sieć. Ciekawy jest też fakt, że dla rozmycia z zakresu $[0.1, 1]$ praktycznie nie zmienia się średnia liczba zdobytych punktów. Świadczyć to może o tym, że ten typ sieci dosyć słabo radzi sobie z parametrami podanymi w sposób rozmyty.

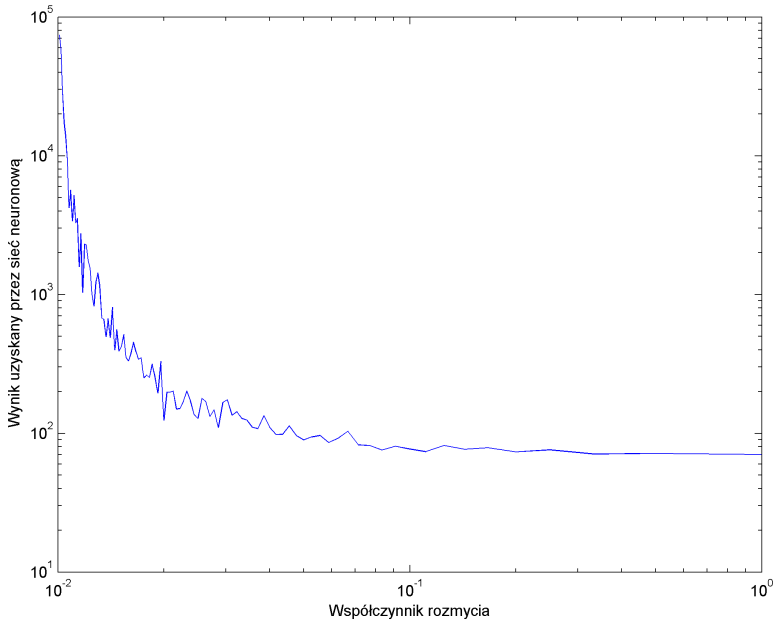
4 Podsumowanie

Uzyskane wyniki świadczą, że przy odpowiednio przygotowanej reprezentacji planszy jest możliwość nauczenia sieci neuronowej o strukturze radialnej reguł grania w proste jednowymiarowe gry komputerowe. Oczywiście dzieje się tak ponieważ rozwiązywany problem posiada prosty opis i jest możliwy podział (klasyfikacja) zbioru odpowiedzi na grupy.

Z powodów ograniczonych zasobów komputerowych nie było możliwe zreplikowanie eksperymentów zaprezentowanych w [1], gdyż próba nauczenia sieci neuronowej za pomocą standardowych metod zajęłaby nierealistycznie dużo czasu a zmniejszenie ilości neuronów w sieci powodowało kompletne nieradzenie sobie jej z tym zadaniem. Z powyższych względów postanowiono ograniczyć przestrzeń problemu i spróbować rozwiązać go rozwiązać na wyższym poziomie abstrakcji. Ciekawym rozwiązaniem mogłoby być jedna

Bibliografia

1. V Mnih, K Kavukcuoglu, D Silver, A Graves, I Antonoglou, D Wierstra, M Riedmiller - Playing Atari with Deep Reinforcement Learning. NIPS 2013 Deep Learning Workshop



Rysunek 4: Liczba uzyskanych punktów przez sieć neuronową w zależności od współczynnika rozmycia danych (1- całkowite rozmycie, 0 - brak rozmycia). Skala wykresu loglog.

2. http://www.omgchrome.com/chrome-easter-egg-trex-game-offline/

3. http://www.pygame.org/hifi.html

4. https://github.com/cybercase/pyradbas

Załączniki

- game.wmv - animacja prezentująca efekt działania sieci neuronowej w trakcie gry