Uczenie perceptronów reguł gry w kółko i krzyżyk

Paweł Szołtysek

Spis treści

1 Zasady gry

Gra w kółko i krzyżyk toczy się między dwoma graczami A i B na planszy składającej się z dziewięciu pól $P_i, i \in 0, \ldots, 8$.

Grę rozpoczyna gracz A, który wybiera dowolne pole, które oznacza znakiem X. Następnie ruch wykonuje gracz B, który wybiera dowolne pole, nie wybrane wcześniej przez żadnego z graczy, i oznacza je znakiem O. W ten sposób gracze grają naprzemiennie, aż do zapełnienia całej tablicy, lub wygranej jednego z graczy.

Wygrana następuje wtedy, gdy któryś z graczy stworzy linię trzech własnych znaków. Natomiast gdy wszystkie pola są zajęte i nie zostaje utworzona linia, gra kończy się remisem.

2 Wykorzystanie perceptronu do gry w kółko i krzyżyk

Perceptron można wykorzystać do wyboru odpowiedniego pola przez gracza. Może to robić na dwa sposoby, określając:

- ruchy zgodne z regułami gry;
- ruchy optymalne, maksymalizujące prawdopodobieństwo wygranej.

Wykonana przez autora implementacja pozwala na określenie ruchów zgodnych z regułami gry.

Wykorzystuje zbiór dziewięciu niezależnych perceptronów $R_i, i \in 0, \ldots, 8$, gdzie każdy z nich odpowiada odrębnemu polu gry P_i. Każdego z nich charakteryzuje dziewięciowymiarowy wektor wejściowy i binarne wyjście określające, czy pole, za które odpowiada dany perceptron, jest zajęte (wtedy wartość wyjściowa powinna wynieść 0), czy też wolne (w tym przypadku 1).

3 Implementacja

Implementacja została wykonana w języku python 2.6. Poza podstawową składnią, wykorzystuje pakiet random, który służy do generowania liczb losowych - wykorzystywany on jest przy losowym generowaniu zbioru uczącego.

3.1 Zasada działania

Program można podzielić na dwie podstawowe części (związane z uczeniem perceptronów oraz sposobem ich wykorzystania) oraz dodatkową (związane z ich wykorzystaniem do gry).

3.1.1 Proces uczenia perceptronu

Proces uczenia zbioru perceptronów wykonywany jest zawsze podczas uruchomienia (kompilacji) aplikacji.

Wywoływany jest przez funkcję learn_tictactoe(inum), gdzie parametr inum wyznacza długość zbioru uczącego.

Funkcja ta tworzy 9 perceptronów $R_i, i \in 0, \ldots, 8$. Dla każdego z nich tworzy losowo sytuacje, które mogą wystąpić na planszy, oraz wyznacza oczekiwane wyjście (tzn. określa, czy pole, za które odpowiada perceptron, jest zajęte). Początkowe wagi dla każdego z wejść są wybierane losowo. Następnie, na podstawie tego zbioru uczącego, wykonywane jest uczenie perceptronu - iterowane są wszystkie przypadki zbioru uczącego, do momentu uzyskania zerowego błędu na zbiorze wejściowym.

3.1.2 Proces *testowania* perceptronu

Druga część aplikacji wykorzystuje nauczone już perceptrony do określania na podstawie zadanego stanu, czy dane pole jest już zajęte czy też nie.

W tym celu funkcja evaluate(situation) przemnaża odpowiednie wektory wartości wejściowych (parametr siatuation) oraz wag, które zostały wyznaczone w procesie uczenia.

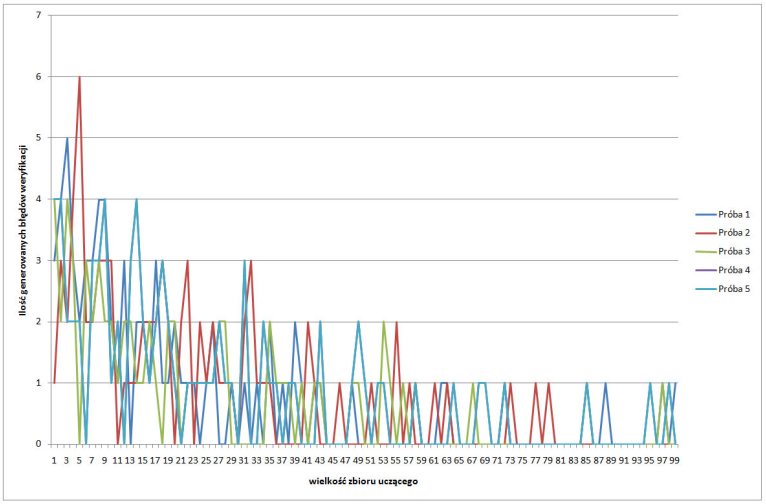
3.1.3 Wykorzystanie zbioru perceptronów do gry

Nauczony zbiór perceptronów można wykorzystać do symulacji gry w kółko i krzyżyk. Zaimplementowane zostały dwa typy rozgrywek - zbiór perceptronów z użytkownikiem oraz zbiór perceptronów z innym (w szegółności z inną długością ciągów uczących) zbiorem perceptronów.

4 Wpływ długości zbioru uczącego na ilość błędów

4.1 Badania dla wskazanej sytuacji

Rysunek 1 przedstawia ilość popełnianych błędów dla wskazanej sytuacji w zależności od długości zbioru uczącego.



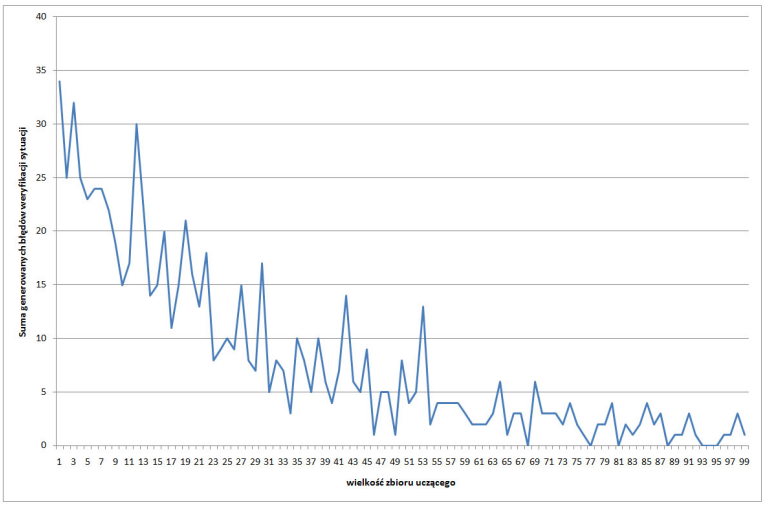
Rysunek 1: Stosunek ilości popełnianych błędów dla wskazanej sytuacji w zależności od długości zbioru uczącego.

Na owym rysunku "próba" oznacza wykonanie jednej iteracji funkcji, która wyznaczała błąd dla stu różnych, **niezależnych** wielkości zbioru uczącego. Wyniki które uzyskano pokazują, że w niektórych przypadkach zaledwie 5 elementów zbioru uczącego wystarczyło, aby zbiór perceptronów był w stanie poprawnie odpowiedzieć, które możliwości ruchów są dozwolone.

Jednak optymalną wartością wielkości zbioru uczącego w tym teście wydaje się być ok. 60 - powyżej tej wartości błędy występują już stosunkowo rzadko. Akceptowalną wartość wielkości tego zbioru osiągamy już na poziomie 25.

4.2 Badania dla sytuacji losowych

Rysunek 2 przedstawia sumę popełnianych błędów dla losowo wybranych 10 sytuacji w zależności od długości zbioru uczącego.



Rysunek 2: Stosunek ilości popełnianych błędów dla losowych 10 sytuacji w zależności od długości zbioru uczącego.

Wyniki otrzymane w tej części potwierdzają te otrzymane dla sztywnej, pojedynczej sytuacji narzuconej z góry. Optymalna wielkość zbioru uczącego wydaje się oscylować w okolicy 60.

5 Wnioski

Uzyskane przez nas wartości wielkości zbioru uczącego wydają się bardzo mała w stosunku do sumy ilości możliwych stanów gry. Należy jednak zauważyć, iż w tym ciągu uczącym przyjęliśmy losowo wybrane sytuacje, które zwykle będą pokrywały w sposób losowy, a przez to stosunkowo jednolity różne stany w jakich może znaleźć się plansza gry. Zwykle, w zastosowaniach praktycznych nie jesteśmy w stanie wybrać takich różniących się w dużym stopniu sytuacji, co powoduje potrzebę wystąpienia większej ilości elementów w zbiorze uczącym.