MSI2 Four-in-a-row

Łukasz Dragan, Piotr Izert

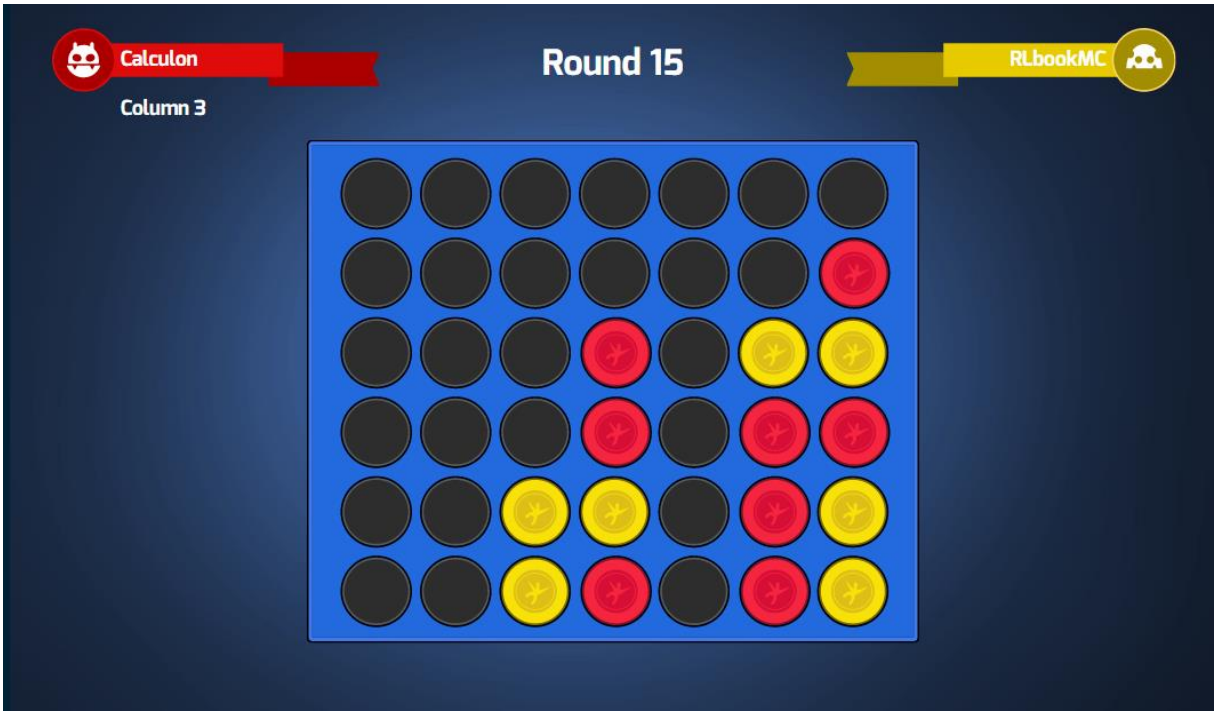
11.06.2016

Temat projektu

Celem projektu jest stworzenie inteligentnego bota do gry Four-in-a-row[1] (zwanej również „Connect Four", po polsku „Czwórki"). Gra jest jedną z wielu udostępnionych w serwisie The AI Games[2], który oferuje uczestnikom możliwość rywalizacji przy implementacji sztucznej inteligencji do gier. W celu wyszkolenia bota użyliśmy zasady uczenia nienadzorowanego.

Zasady gry

W grze uczestniczy dwóch graczy. Gra odbywa się na planszy 7x6. Gracze na zmianę kładą po jednym kamieniu o kolorze przypisanym do gracza na planszy. Położenie kamienia odbywa się poprzez wybór kolumny, w której kamień zostaje umieszczony w niezajętym polu o najniższym numerze wiersza. Wygrywa ten, kto pierwszy ułoży 4 swoje kamienie w jednym rzędzie: pionowo, poziomo lub na ukos.



Rysunek 1 Wizualizacja gry w serwisie The AI Games

[1] https://en.wikipedia.org/wiki/Connect_Four
[2] http://theaigames.com/competitions/four-in-a-row

Metodyka

Szkolenie bota odbywa się za pomocą trenowania sieci neuronowej przy użyciu algorytmu ewolucyjnego. Osobnik populacji odpowiada wektorowi wag sieci neuronowej.

Ewolucja

W algorytmie wykorzystujemy selekcję turniejową, krzyżowanie i mutację. Wykonujemy *n_i* iteracji algorytmu ewolucyjnego dla populacji o wielkości *n*. Rozgrywane jest 2*n* turniejów o stałej liczbie uczestników *t*. Zwycięzcy dwóch kolejnych turniejów są ze sobą krzyżowani. Zasada krzyżowania: do osobnika potomnego trafia gen *i* rodzica A z prawdopodobieństwem *p_c*. W przeciwnym razie trafia do niego gen *i* rodzica B. Następnie dla każdego genu osobnika potomnego dodawana jest z prawdopodobieństwem *p_m* zmienna losowa pochodząca z rozkładu gaussowskiego.

Zasady turnieju:

Wśród osobników wybranych do turnieju rozgrywane są pojedynki w grę Four-in-a-row na zasadzie „każdy z każdym”. Zwycięzca największej liczby pojedynków jest zwycięzcą turnieju. Pojedynek odbywa się poprzez serię kolejnych ruchów wykonywanych przez rywalizujące boty. Każdy z botów otrzymuje wektor wag dla sieci neuronowej, którą wykorzystuje do wyznaczenia kolumny na podstawie aktualnego stanu planszy.

Sieć neuronowa

Do wyboru optymalnego ruchu w aktualnej sytuacji boty używają sieci typu Wielowarstwowy Perceptron. Badano różne warianty topologii sieci, w testach użyto sieci z jedną warstwą ukrytą z 10 neuronami oraz z dwiema warstwami ukrytymi po 10 neuronów.

Wejściem sieci są 42 liczby, które odpowiadają stanom każdego z 7x6 pól planszy. Wyjściem sieci jest 7 neuronów – każdy reprezentuje indeks kolumny, w której należy dokonać ruchu, wybierany jest neuron z najsilniejszym sygnałem. Jeżeli ruch zwrócony przez sieć jest w danej sytuacji niedopuszczalny, wybierany jest inny ruch o największej wartości sygnału.

Użyte biblioteki

Do wykonania programu testującego algorytmy użyliśmy standardowej biblioteki Java. Jedyną biblioteką zewnętrzną, którą wykorzystaliśmy jest Neuroph[3]. Biblioteka ta jest napisana w języku Java i umożliwia konstrukcję sieci neuronowej o wymaganej topologii. Kod źródłowy biblioteki jest otwarty i udostępniony na licencji Apache, co ułatwiło wykorzystanie i zrozumienie działania biblioteki.

Biblioteka Neuroph zawiera takie klasy, jak neuron, warstwa neuronów, czy sieć, oraz potrzebne metody. Dzięki temu w czytelny sposób można było zdefiniować potrzebne rodzaje sieci, a następnie ustawić wagi połączeń między neuronami na podstawie genotypu danego osobnika, podać sygnał wejściowy i odczytać odpowiedź sieci.

[3] http://neuroph.sourceforge.net

Testy

W celu wybrania najlepszych parametrów naszej metody wykonaliśmy serię testów. Parametry wejściowe testów:

1. *n_i*=1000, *n*=100, *t*=7, *p_m*=0.015, *p_c*=0, sieć neuronowa: z jedną warstwą ukrytą, wejście sieci: plansza z polami o wartościach: 0, 1, 2
2. *n_i*=1000, *n*=100, *t*=7, *p_m*=0.015, *p_c*=0.35, sieć neuronowa: z jedną warstwą ukrytą, wejście sieci: plansza z polami: 0, 1, 2
3. *n_i*=1000, *n*=100, *t*=7, *p_m*=0.015, *p_c*=0.35, sieć neuronowa: z jedną warstwą ukrytą, wejście sieci: plansza z polami: 0, -1, 1
4. *n_i*=1000, *n*=100, *t*=7, *p_m*=0.015, *p_c*=0, sieć neuronowa: z dwoma warstwami ukrytymi, wejście sieci: plansza z polami: 0, 1, 2
5. *n_i*=1000, *n*=100, *t*=7, *p_m*=0.015, *p_c*=0.35, sieć neuronowa: z dwoma warstwami ukrytymi, wejście sieci: plansza z polami: 0, 1, 2
6. *n_i*=1000, *n*=100, *t*=7, *p_m*=0.015, *p_c*=0.35, sieć neuronowa: z dwoma warstwami ukrytymi, wejście sieci: plansza z polami: 0, -1, 1

Co 10 iteracji wyłaniamy w turnieju, w którym udział bierze cała populacja, najlepszego osobnika. Wykonuje on 1000 pojedynków ze specjalnie przygotowanym botem.

Zasada działania bota testowego

Wykonuje losowy ruch, chyba że na planszy znajduje się pionowy układ trzech kamieni o tym samym kolorze oraz pole ponad nimi jest wolne. Wtedy, jeżeli kamienie są w kolorze bota i może on wygrać grę, kładzie swój kamień i wygrywa. W przeciwnym wypadku blokuje on ruch przeciwnika.

Bot, mimo, że jest raczej prymitywny, wygrywa większość gier przeciwko botowi wykonującemu jedynie ruchy losowe.

Porównanie wyników

Dla każdego z testów otrzymujemy 100 wyników cząstkowych oznaczających część gier wygranych z botem testowym (wartości od 0 do 1):

1. 0.139 0.051 0.18 0.168 0.223 0.062 0.062 0.074 0.166 0.227 0.137 0.326 0.153 0.166 0.139 0.26 0.29 0.193 0.218 0.128 0.175 0.161 0.222 0.185 0.2 0.144 0.249 0.208 0.206 0.165 0.248 0.201 0.3273333333333333 0.313 0.286 0.262 0.122 0.286 0.386 0.291 0.265 0.294 0.282 0.354 0.221 0.273 0.331 0.206 0.267 0.405 0.334 0.24 0.234 0.271 0.188 0.353 0.348 0.434 0.271 0.26 0.246 0.31 0.25 0.227 0.328 0.289 0.282 0.264 0.219 0.229 0.252 0.287 0.403 0.375 0.345 0.361 0.278 0.352 0.272 0.295 0.272 0.225 0.424 0.202 0.113 0.399 0.351 0.458 0.306 0.304 0.409 0.432 0.377 0.376 0.31 0.376 0.314 0.413 0.354 0.233

2. 0.063 0.093 0.149 0.069 0.064 0.211 0.092 0.086 0.142 0.284 0.188 0.291 0.306 0.181 0.21 0.144 0.339 0.512 0.534 0.483 0.472 0.547 0.505 0.683 0.541 0.515 0.591 0.498 0.671 0.405 0.412 0.686 0.645 0.73 0.579 0.658 0.552 0.585 0.635 0.599 0.534 0.607 0.606 0.548 0.568 0.576 0.662 0.607 0.588 0.632 0.676 0.594 0.637 0.705 0.62

0.709 0.658 0.705 0.653 0.637 0.643 0.568 0.537 0.631 0.561 0.606
0.455 0.575 0.518 0.49 0.492 0.533 0.463 0.507 0.484 0.587 0.583 0.53
0.522 0.482 0.53 0.521 0.501 0.498 0.407 0.565 0.515 0.501 0.505
0.581 0.582 0.546 0.574 0.565 0.565 0.596 0.538 0.619 0.528 0.571
0.558 0.435 0.493

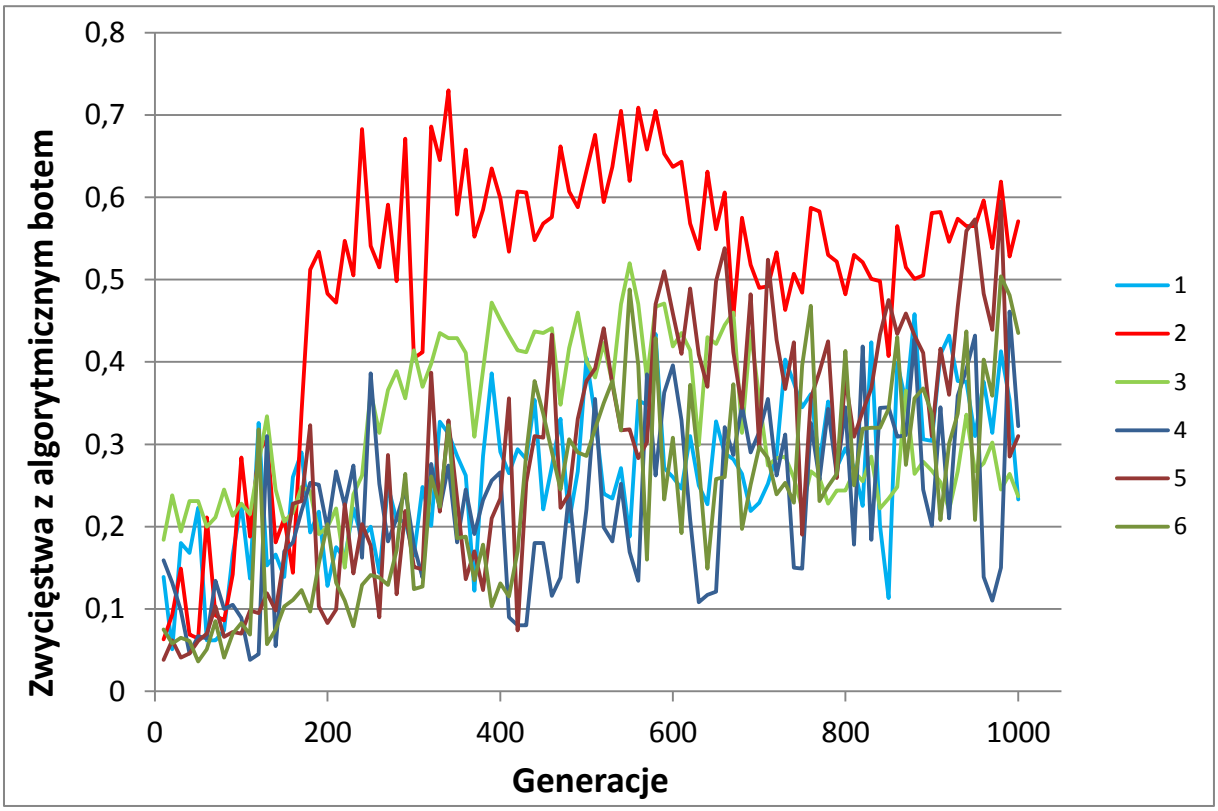
3. 0.184 0.238 0.194 0.231 0.231 0.2 0.211 0.245 0.213 0.228 0.215 0.277
0.334 0.245 0.206 0.217 0.248 0.243 0.191 0.199 0.222 0.15 0.239
0.262 0.364 0.314 0.366 0.389 0.356 0.414 0.37 0.398 0.435 0.429
0.429 0.411 0.309 0.39 0.472 0.451 0.432 0.414 0.412 0.437 0.435
0.441 0.348 0.417 0.46 0.401 0.381 0.423 0.374 0.47 0.52 0.47 0.38
0.467 0.471 0.419 0.435 0.414 0.299 0.43 0.422 0.445 0.46 0.314 0.437
0.356 0.274 0.283 0.285 0.259 0.222 0.267 0.259 0.228 0.244 0.244
0.267 0.255 0.285 0.222 0.234 0.248 0.365 0.264 0.279 0.268 0.254
0.22 0.267 0.336 0.267 0.277 0.302 0.245 0.264 0.237

4. 0.159 0.132 0.098 0.046 0.067 0.064 0.134 0.1 0.105 0.089 0.038 0.045
0.31 0.055 0.17 0.181 0.219 0.253 0.251 0.197 0.267 0.225 0.274 0.162
0.386 0.252 0.182 0.21 0.245 0.177 0.138 0.276 0.221 0.274 0.181
0.245 0.191 0.232 0.256 0.266 0.09 0.08 0.08 0.18 0.18 0.116 0.138
0.243 0.133 0.221 0.355 0.199 0.182 0.252 0.169 0.134 0.385 0.262
0.363 0.396 0.329 0.209 0.108 0.117 0.121 0.321 0.287 0.347 0.29
0.315 0.355 0.262 0.312 0.15 0.149 0.326 0.257 0.343 0.263 0.345
0.178 0.419 0.184 0.344 0.345 0.309 0.311 0.428 0.245 0.201 0.345
0.21 0.359 0.391 0.432 0.139 0.11 0.15 0.461 0.322

5. 0.038 0.062 0.041 0.046 0.061 0.07 0.103 0.066 0.072 0.07 0.098 0.095
0.119 0.098 0.163 0.228 0.231 0.323 0.103 0.083 0.099 0.227 0.143
0.203 0.177 0.09 0.287 0.118 0.219 0.151 0.148 0.387 0.218 0.329
0.234 0.136 0.17 0.123 0.21 0.234 0.356 0.074 0.254 0.31 0.308 0.433
0.223 0.24 0.331 0.377 0.392 0.441 0.373 0.317 0.318 0.283 0.302 0.47
0.51 0.459 0.41 0.489 0.409 0.37 0.498 0.538 0.412 0.344 0.482 0.296
0.524 0.427 0.367 0.424 0.19 0.356 0.388 0.425 0.259 0.38 0.309 0.34
0.368 0.433 0.475 0.434 0.459 0.432 0.411 0.31 0.416 0.36 0.467 0.559
0.573 0.483 0.439 0.595 0.285 0.31

6. 0.075 0.058 0.065 0.061 0.036 0.051 0.085 0.041 0.07 0.083 0.069
0.318 0.057 0.075 0.103 0.111 0.123 0.097 0.155 0.203 0.132 0.11
0.079 0.129 0.141 0.138 0.129 0.171 0.264 0.124 0.127 0.261 0.224
0.323 0.186 0.188 0.135 0.178 0.103 0.131 0.115 0.17 0.289 0.377
0.337 0.291 0.245 0.306 0.29 0.286 0.319 0.35 0.377 0.317 0.488 0.396
0.16 0.429 0.233 0.308 0.192 0.372 0.276 0.149 0.258 0.26 0.373 0.197
0.251 0.297 0.283 0.239 0.253 0.229 0.396 0.468 0.231 0.249 0.264
0.41333333333333333 0.25 0.319 0.32 0.32 0.343 0.43 0.275 0.356 0.368
0.337 0.208 0.301 0.337 0.437 0.208 0.403 0.359 0.504 0.481 0.435

Wyniki przedstawiono również na wykresie:



Wnioski

Wykres pokazuje, że najlepsze wyniki dała metoda korzystająca z sieci z jedną warstwą ukrytą i używająca krzyżowania osobników. Dla każdej użytej kombinacji parametrów można zaobserwować, iż po ok 500 iteracjach wykres stabilizuje się.

Jedną z trudności okazał się brak jawnej funkcji celu dla algorytmu ewolucyjnego. Zaprezentowane wykresy nie są monotoniczne, ponieważ nie optymalizowaliśmy gry przeciw botowi testowemu, a przeciw każdemu możliwemu przeciwnikowi.

Optymistycznym jest fakt, iż w obrębie populacji dało się zaobserwować zróżnicowanie jakości osobników. Najlepszy osobnik w turnieju „każdy z każdym” dla 99 pojedynków otrzymywał średnio 95-99 zwycięztw.

Wynik działania algorytmu radzi sobie lepiej w pojedynku z opisanym botem testowym niż bot wykonujący jedynie ruchy losowe. Mimo to uzyskany gracz nie jest obiektywnie silny. Być może sama metoda uczenia nienadzorowanego ma swoje ograniczenia, których nie da się przekroczyć wyłącznie przez modyfikacje jej parametrów.

Źródła

http://neuroph.sourceforge.net

http://theaigames.com/competitions/four-in-a-row

https://en.wikipedia.org/wiki/Connect_Four