Sudoku solver

Katarzyna Król

25 stycznia 2020

1 Opis

Projekt ma na celu rozwiązanie instancji zagadki sudoku za pomocą sieci neuronowych. Solver otrzymuje prawidłową instację problemu w postaci macierzy 9x9 wypełnionej cyframi, gdzie 0 – oznacza pole do uzupełnienia, a pozostałe cyfry są ustalone. Wynik polega na zastąpieniu zer innymi cyframi, tak, aby spełniały założenia poprawnego rozwiązania.

Zastosowałam następujące techniki trenowania i testowania:

1.1 Trenowanie

Wejście: (quizzes, solutions) z kaggle.

1.1.1 Trenowanie krótkie

Ustawienie features=quizzes i labels=solutions
Typowe trenowanie - przejście przez cały zbiór treningowy za pomocą batchy w kilku epokach.

1.1.2 Trenowanie długie

W tym podejściu zbiór features zmienia się. Początkowo features=solutions i labels=solutions, dzięki czemu sieć uczy się, że uzupełnione pola mają pozostać bez zmian. Następnie usuwam po jednej cyfrze z każdego feature, po czym ponownie trenuję sieć. Tworzę w ten sposób coraz mniej uzupełnione sudoku, które sieć otrzymuje do trenowania (pola do usuwania wybieram spośród tych z quizzes, aby zapewnić cały czas jednoznaczność rozwiązania).

1.2 Testowanie

1.2.1 Testowanie krótkie

Typowe testowanie - sieć przewiduje prawdopodobieństwa wystąpień każdej cyfry na każdym polu. Rozwiązaniem jest wybranie najbardziej prawdopodobnej cyfry na każdej pozycji.

1.3 Testowanie długie

Uzupełnianie zagadki krok po kroku:

- obliczenie za pomocą sieci prawdopodobieństw wystąpienia każdej cyrfy na każdej pozycji
- spośród pustych pól wybranie pozycji z najbardziej prawdopodobną cyfrą
- wstawienie tej cyfry do zagadki
- uzyskana w ten sposób zagadka ma o jedno puste pole mniej i jest ponownie ładowana do sieci (powrót do punktu pierwszego)

2 Metody sztucznej inteligencji

Zastosowałam sieci:

- Sieć konwolucyjna z warstwami:
  - warstwa konwolucyjna
  - batch normalization
  - flatten
  - fully connected
- Sieć gęsta bez warstw ukrytych.

Funkcja straty - softmax cross entropy loss.

3 Narzędzia

Projekt napisałam w języku python. Skorzystałam z bilbliotek numpy i mxnet. Do trenowania i testowania użyłam danych dostępnych na kaggle.

4 Wkład własny

- Przygotowanie danych
- Napisanie batch iteratora
- Sterowanie trenowaniem
- Obliczenie, ile pustych pól w danych testowych zostało poprawnie odgadniętych

Do skonstruowania i trenowania sieci użyłam metod z biblioteki mxnet.

5 Wyniki

5.1 Moje wyniki

Podany wynik oznacza ile spośród wszystkich pustych pól zostało uzupełnionych poprawnie.

- W przypadku długiego treningu najlepiej sprawdzała się sieć gęsta:
  - Testowanie krótkie - wynik = 0.8320987654320988
  - Testowanie długie - wynik = 0.8506172839506173
- W przypadku krótkiego treningu lepsze wyniki osiągała sieć konwolucyjna:
  - Testowanie krótkie - wynik = 0.7276596
  - Testowanie długie - wynik = 0.7276596

5.2 Wyniki innych badaczy oraz źródła

- "99.7% accuracy" (https://www.kaggle.com/dithyrambe/neural-nets-as-sudoku-solvers)
- "Total Accuracy 1345/1568 = 0.86" (hhttps://github.com/Kyubyong/sudoku/blob/master/README.md)
- "Test accuracy on 1000 games was 0.99." (https://towardsdatascience.com/solving-sudoku-with-convolution-neural-network-keras-655ba4be3b11)