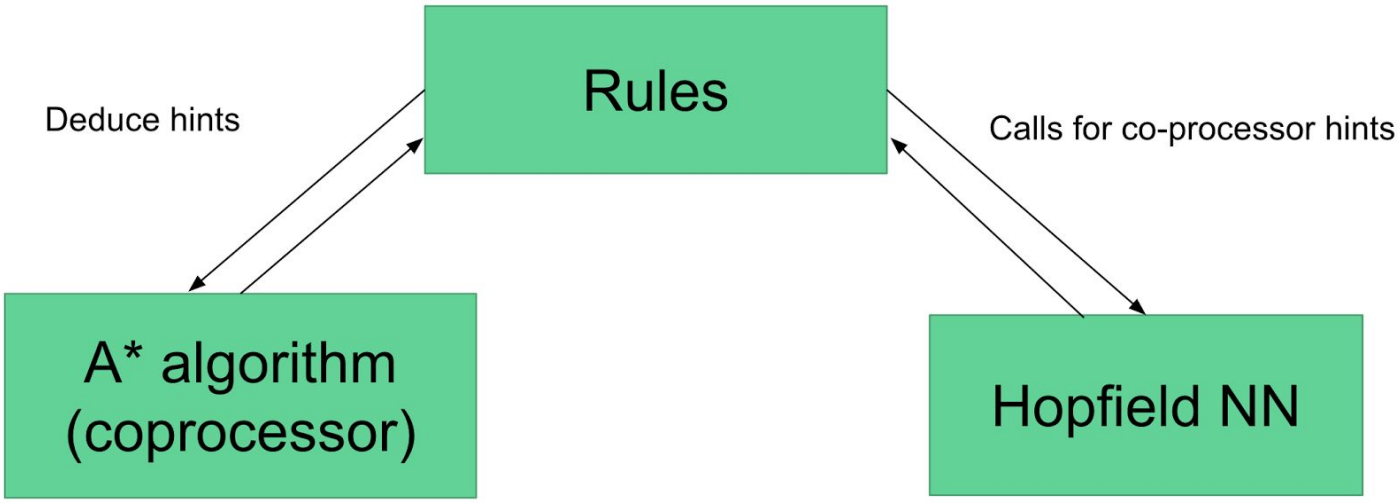
Sudoku Solver

Kumari Anamika Sharaf, Saurabh Marathe, and William Nguyen

December 19, 2016



Abstract

Sudoku is a puzzle game with a 9x9 board separated into nine 3x3 boxes. The rules for sudoku dictate that each row, column, and box need to be filled with values from 1 to 9 and no value should repeat itself. Although the rules are simple, it requires a lot of thinking to narrow down possible options for each cell and getting to the final solution.

Most previous solutions utilize brute force to analyze all possible paths to fill the sudoku board. However, this approach can be time consuming for more complicated boards. Our solution is to utilize artificial intelligence and machine learning with knowledge base to find an optimal solution. Here, we will use the A* search algorithm for artificial intelligence and a Hopfield neural network for machine learning, and will integrate everything with a rule-based system for knowledge representation.

Inspiration

As a puzzle, sudoku is very similar to grid-based games such as tic-tac-toe and chess. These previous solutions utilize the search algorithm known as minimax to determine the best next move to make. Search algorithms are great in grid-based games since they explore all possible paths before making a move. Unfortunately, sudoku is a one-player game and is not a classification problem. This means we need an alternative approach to solve sudoku.

Our decision to use the A* search algorithm comes from the algorithm's use of heuristics to make better decisions while searching. These heuristics allow the A* search algorithm to run faster while still providing 100% accuracy.

Unlike other artificial neural networks, the Hopfield neural network has associative memory as its property. This means its neurons consider both its input and inputs from other neurons to make decisions. For more complex boards, this means the Hopfield neural network may require several iterations before it can completely solve the puzzle.

Since both the A* search and Hopfield neural network are both good algorithms to solve sudoku, we want to tie them together using a rule-based system. In the instances where the Hopfield neural network may be confused about which values go into a particular cell, the rule-based system will provide extra hints with the help of the A* search algorithm to guide the Hopfield neural network into solving sudoku successfully.

Implementation

A* Search

The search algorithm we will use is an A* search algorithm to solve sudoku. Each node of the search algorithm will be the original sudoku board with various empty cells filled during the search. To decrease the search space and decrease unnecessary redundant search paths (i.e. filling order is not important) the algorithm will always expand a board by filling only one empty cell rather than all possible empty cells. This empty cell that will be filled in each expansion is one of the empty cells with the least amount of possible values.

Unlike a search algorithm such as depth-first search, the A* search algorithm will be faster since it uses heuristics to make better decisions while searching. We've decided the heuristics will be determined by the amount of uncertainty of the sudoku board. This uncertainty is the total amount of possible values for each of the empty cells. These possible values are narrowed down by removing values already filled in the cell's domain: the row, column, and box that the cell resides in. The more unfilled a board is and the more possible values a cell has will cause the uncertainty of the sudoku board to increase. By using this as our heuristic, A* search will first expand states where the possible values of cells are narrowed down the furthest.

Hopfield Neural Networks

The Hopfield Neural Network is a neural network with a graph $G = (U, C)$. It uses associative memory technique to solve a problem. Associative memory here is nothing but the memory Hopfield Neural Network maintains by giving the output of first iteration as an input for second iteration along with original input and henceforth.

Our Hopfield neural network has a 9x9x9 $(i \times j \times k)$ matrix of 729 neurons where i represents row, j represents column and k represents the value that cell (i,j) will hold.

In the binary representation, the rules for the solution of Sudoku are:

1. $V(i,j,k) = 1$ for the set locations defining the particular puzzle where the cells are filled
2. $V(i,j,k) =$ 0 or 1 for all other i, j, k
3. $\sum_i V(i,j,k) = 1$ for all j, k
4. $\sum_j V(i,j,k) = 1$ for all i, k
5. $\sum_k V(i,j,k) = 1$ for all i, j

6. $\sum_{i,j} V(i,j,k) = 1$ for all k, with the sum on i and j taken over one of the 3 × 3 i,j squares

The above equations are determined for many cells after two iterations of updating the energies of each neuron. The energy is the sum of the inputs to a neuron. After two iterations of updating the energies, if there are any neurons (i,j,k) that have equations 3, 4, 5, and 6 as zero, then the neural network knows that cell i,j is the cell that has k as a possible value. This is how updating energies in the system helps the Hopfield neural network to determine the cells to be filled along with the possible value to be put in that cell. It may happen that a cell i,j may have more than one possible value. This is where HNN requires hints.

Please note that we require two calls to the energy update function because

1. The first iteration initializes the output for all neurons linearly with respect to its input.
2. The second iteration helps bring the output of other neurons in the same domain, as input to each neuron.

Rule-Based System

Our knowledge representation will be in terms of a rule-based system. Here, the rule base integrates both the A* search algorithm and the Hopfield neural network. The A* search algorithm behaves as a co-processor and helps the Hopfield neural network solve Sudoku successfully.

The A* search algorithm solves sudoku with 100% accuracy, but it takes longer time to solve as compare to the Hopfield neural network. In case Hopfield neural network gets confused while deciding which value to pick for a particular cell, the rule-based system will be fired to assist the Hopfield neural network. It will provide hints given by the A* search algorithm to help Hopfield achieve its goal.

The hint provided will be a value to one of the uncertain cells. The cell in particular that will be filled is the cell with the highest number of common possible values that it shares with other uncertain cells. This probability based rule ensures that we reduce the amount of confusion in the Hopfield neural network while just giving a value for one cell.

Results

For a sample sudoku input, we have provided the following board:

```
sudoku_puzzle = [
        [8,9,1,2,7,4,5,6,0],
        [6,0,3,1,8,5,9,0,0],
        [4,5,7,6,3,9,0,0,0],
        [5,0,6,4,1,7,2,3,0],
```

```
        [7,4,2,9,0,3,8,1,0],
        [3,1,0,0,2,6,0,5,0],
        [9,3,8,5,4,0,6,7,0],
        [1,6,4,7,9,0,3,2,0],
        [0,7,5,3,6,1,4,9,0]
    ]
```

The program first runs it through the Hopfield neural network and the following semi-completed board is the output of this first iteration.

```
[8, 9, 1, 2, 7, 4, 5, 6, 3]
[6, 2, 3, 1, 8, 5, 9, 4, 0]
[4, 5, 7, 6, 3, 9, 1, 8, 0]
[5, 8, 6, 4, 1, 7, 2, 3, 9]
[7, 4, 2, 9, 5, 3, 8, 1, 6]
[3, 1, 9, 8, 2, 6, 7, 5, 0]
[9, 3, 8, 5, 4, 2, 6, 7, 1]
[1, 6, 4, 7, 9, 8, 3, 2, 0]
[2, 7, 5, 3, 6, 1, 4, 9, 8]
```

The semi-completed board still has 4 uncertain cells in the final column, which the Hopfield neural network has provided possible values to fill in.

```
Neural Network confused about following cells, and their possible
values are:-
Cell 1,8 has following possible hints
2 4 7
Cell 2,8 has following possible hints
1 2 8
Cell 5,8 has following possible hints
4 7 9
Cell 7,8 has following possible hints
5 8
```

At this point, the program determines the best next cell to fill in order to have the Hopfield neural network continue solving the puzzle. This cell is the one with the highest commonly shared possible values with other cells. In this case, it is cell (1,8), also known as cell 17.

```
The cell chosen is 1,8, with commonality index as 3
17
```

Next, the rule-based system will fire the A* search algorithm, the co-processor, to fetch the correct value to be filled in cell (1,8).

```
This is a hint provided by A* solution
7 is the value to be put in 1,8
```

The Hopfield neural network will run again with the updated semi-solved sudoku board from the first iteration and the hint provided by the A* search algorithm for the cell with the highest commonality index.

```
Output board is:-
[8, 9, 1, 2, 7, 4, 5, 6, 3]
[6, 2, 3, 1, 8, 5, 9, 4, 7]
[4, 5, 7, 6, 3, 9, 1, 8, 2]
[5, 8, 6, 4, 1, 7, 2, 3, 9]
[7, 4, 2, 9, 5, 3, 8, 1, 6]
[3, 1, 9, 8, 2, 6, 7, 5, 4]
[9, 3, 8, 5, 4, 2, 6, 7, 1]
[1, 6, 4, 7, 9, 8, 3, 2, 5]
[2, 7, 5, 3, 6, 1, 4, 9, 8]
```

As we can see, the neural network is confused about values for 4 cells. But giving value for cell 1,8 (or cell 17), helped the neural network solve the values for all the other cells. Now, the important thing to observe is that giving a value to cell 17 is important. If instead of cell 17, we give value of some other cell as a hint, then the neural network may take longer time to solve. The reason for this is because the hint may not help narrow down the possible values for the other uncertain cells. Because cell 17 has the most values common to all the other cells, giving a value to cell 17 reduces the number of possible values for other cells. Hence, our rule of supplying value of cell with highest number of common possible values (known as the commonality index) is useful.

Now, the problem has been solved by the Hopfield neural network in two iterations. No matter how many iterations the Hopfield neural network has run, A* search will only run once since it solves the puzzle correctly.

Conclusion

The A* search algorithm solves the sudoku puzzle exhaustively, but provides 100% accuracy. Because of its exhaustive search, it can take a long time to solve more complicated sudoku puzzles, whereas the Hopfield neural network runs fast but needs help to figure out the correct possible values for a few of the confused cells.

As we can see in our results, the program is able to completely solve the sudoku puzzle in a few steps rather than through an exhaustive search. We were able to implement artificial intelligence, machine learning and knowledge representation successfully. Some things we noticed is that for harder sudoku problems, the Hopfield neural network takes a larger number of iterations to solve the puzzle whereas smaller problems comparatively take fewer iterations to finish.

Distribution of Work

For this project, the code work was split among each member. The A* search algorithm was written by William Nguyen. The Hopfield neural network was written by Saurabh Marathe. The rule-based knowledge representation was written by Kumari Anamika Sharaf. Each of the code was reviewed by other team members to ensure that the code is efficient, correct, and is able to be integrated properly. Final outputs and results were verified by each member, and all the documentation for this project was contributed by all team members by working together on Google Docs.

References

- Berggren, P., & Nilsson, D. (2012). A study of Sudoku solving algorithms. Royal Institute of Technology, Stockholm.
- Clementis, L. (2015). Study of Game Strategy Emergence by Using Neural Networks. Information Sciences and Technologies, 7(3/4), 26.
- Hopfield, J. J. (2008). Searching for memories, Sudoku, implicit check bits, and the iterative use of not-always-correct rapid neural computation. Neural computation, 20(5), 1119-1164.
- Kanakia, A., & Klingner, J. Methods for Solving Sudoku Puzzles. CSCI-5454, CU Boulder.
- Mladenov, V., Karampelas, P., Pavlatos, C., & Zirintsis, E. (2011). Solving sudoku puzzles by using hopfield neural networks. Proc. of ICACM, 11, 174-179.