Nash

User Documentation

Contents

1 About Nash

Nash is a simple command line tool which searches for stable strategy choices, so called *Nash equilibria*, in games of two players. These equilibria determine strategies for each player such that nobody has the incentive to use a different strategy instead. They may not, however, capture the best strategies overall.

The program is designed with efficiency in mind, such that the running time is plausible even in situations where players have many different actions to choose from.

2 Games

To represent a game of two players, Nash uses a standard model from game theory called a *bimatrix game*. In this model, both players have their own set of *actions* to choose from. These sets may differ or even have different sizes.

Both players choose one action to play. Each possible outcome yields some *payoff* to players; the greater the payoff, the better the outcome was for the given player. The game is only played once.

Players can develop *strategies* by assigning each of their actions a probability with which they will choose it. If strategies of both players are known, it is possible to calculate the *expected payoff* of each player, which is defined simply as the expected value of the payoff.

2.1 Nash equilibria

Naturally, both players want to choose their strategy in such a way that their expected payoff is maximized. They traditionally do not know the other players' strategy. However, if they do, they can adapt themselves to it by using a *best response* strategy.

Such best response may, in turn, cause the other player to create their own best response to it, resulting in a chain reaction of changing strategies. Once this process converges, a Nash equilibrium is reached.

In other words, Nash equilibria are states where each player's strategy is a best response to the other player's strategy. Therefore is does not make sense for them to deviate from their strategies. Every game has at least one Nash equilibrium.

Note that a Nash equilibrium does not guarantee the optimal outcome. If both players agreed on changing their strategies, they could achive better results. A Nash equlibrium only ensures the best outcome under the assumption that the other player will not change their strategy.

2.2 Matrix representation

The described model of a two player game can be captured by two $m \times n$ matrices M and N, where m is the number of actions of player 1, n is the number of actions of player 2. M describes payoffs for player 1, N for player 2. Action played by player 1 determines the row, action played by player 2 determines the column where the payoff for this outcome is found.

A famous bimatrix game is the **Prisoner's dilemma**. This game is represented by the following matrices:

$$M = \begin{bmatrix} -1 & -4 \\ 0 & -3 \end{bmatrix} \qquad N = \begin{bmatrix} -1 & 0 \\ -4 & -3 \end{bmatrix}$$

First row and first column correspond to the action "stay silent", second row and second column to the action "testify".

The only Nash equilibrium of this game is for both players to testify. Player 1 therefore assigns probability 0 to his first action and probability 1 to his second action, which is captured by a strategy vector $x = (0, 1)^\top$. The strategy vector of player 2 is the same, $y = (0, 1)^\top$. The resulting expected payoff for both players will be -3.

As stated before, this is not the optimal outcome. If both players agreed on strategies $x = y = (1, 0)^\top$, their expected payoff would be -1. However, this is not an equilibrium, because any player can betray and get an even better outcome of 0.

3 Usage

Running `nash --help` will print out a basic usage guide. This document describes the same in more detail.

The syntax of the command is `nash [-n NUM] [-s SEP] [FILE]`.

3.1 Matrix input

Payoff matrices can be defined in a text file or entered using standard input.

An $m \times n$ matrix is defined on m consecutive lines. Each line contains n decimal payoff values separated by whitespace. The separator can be customized with options. The two matrices are separated by one empty line.

The matrices from Prisoner's dilemma can be formatted as follows:

```
-1  -4
 0  -3

-1   0
-4  -3
```

Another possibility, which showcases decimal payoffs and non-square matrices. Player 1 has only 1 action to play.

```
-1  1.2

2.4 -2
```

3.2 Options

- `-n` changes the number of equilibria that Nash will search for. By default, only one equilibrium is found. When a value greater than 1 is entered, the search process will be repeated multiple times, possibly yielding other equilibria. If the desired number of equilibria cannot be found, a smaller number of them will be printed.

  Failure to find more equilibria does not imply that no more equilibria exist. Nash can only find at most $m + n$ equilibria.

- `-s` customizes the string used to separate individual payoffs in matrices. If matrix contains multiple repetitions of the separator, it is treated as one. Separator between rows cannot be customized.

3.3 Examples

Examples of games can be found in the `example/` directory.

- Simply running `nash` without any additional arguments prompts for matrices on standard input. After second empty line, Nash starts looking for an equilibrium.

- `nash example/prisoners_dilemma` reads matrices from one of the example files.

- `nash < example/prisoners_dilemma` does the same.
- `nash -n 100 example/close_far` lists all equilibria that Nash can find for the given game.
- `nash -s "_"` allows underscore to be used instead of whitespace in source file or on standard input.

4 Links

- Nash: `https://github.com/kulisak12/Nash`