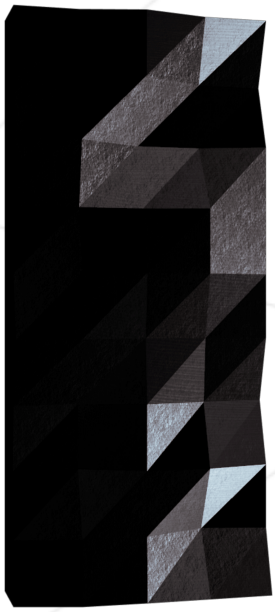


Reverse Game of Life

Learn how to predict the past!

Gaetan gaetan@42.us.org
Sasha sasha@42.us.org

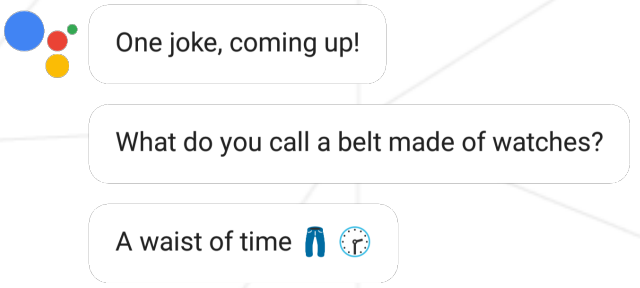
Summary:
The goal of this project is to understand how machine learning works through the creation of your own ML algorithm.

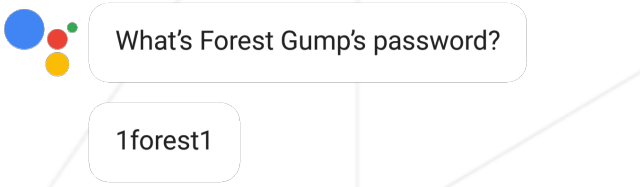
Contents

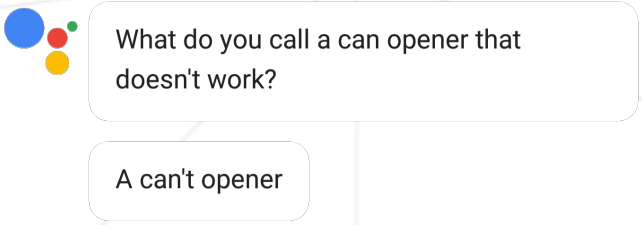
Chapter I

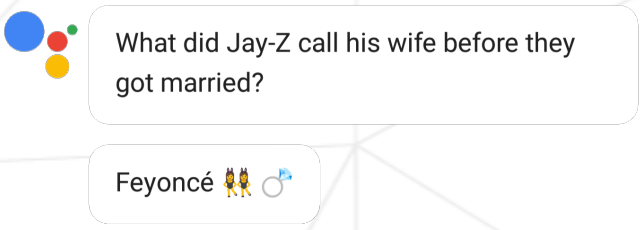
Foreword

If you ever find yourself sad, Google Assistant is always here to cheer you up:









Chapter II

Introduction

Have you ever wondered if there is life in other dimensions? What is it like to live in a 2D plane? And if you find yourself in such place, how can you survive?

Maybe you cannot put your mind in a 2D simulation to experience that (as for now), but John Horton Conway can help you to find the answers in his Game of Life. Conway's Game of Life represents a 2D board of cells, where each cell has 2 possible states: dead and alive. The game represents one of the ways how life can exist in the 2D environment, and it produces fascinating results.

There are only 4 simple rules to this game:

1. Any live cell with fewer than two live neighbors dies, as if caused by underpopulation.
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by overpopulation.
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

Play Conway's Game of Life here

A survey of lifeforms

Chapter III

Goals

Today we will set time backward in Conway's Game of Life. Can you predict what was in the past?

In this project, you will develop an algorithm that will predict a board of the Game of Life based on the board a few life cycles after. For that, you want to use machine learning! "But those libraries for machine learning look so confusing!" you say. "Instead, I will write my own algorithm!"

Throughout this project, you will learn how machine learning works and how you can implement it without the usage of external libraries.

Chapter IV

General instructions

The program must produce predictions of the start boards in Game of Life based on the stop boards.

Use machine learning approach.
The program must be written in Python.

You must write the algorithm for learning and predicting.
Train your program on the test data.
Save your models after training, so the program can predict boards without running training every time.
Display a log during the execution of the program.

Chapter V

Mandatory part

The project must be written in Python.

Your program must:

- have training process and predicting process.
- be trained on the training dataset that we provide, but you can create additional training games.
- be able to save the models for prediction that were created during training process.
- be able to load the models to start prediction process without prior training.
- produce submission.csv file of predicted boards.
- make predictions based only on the machine learning algorithm that you wrote.
- display a log.

Examples of the program execution:

```
$> python train.py train.csv
Opened train.csv
--- Training ---
Training #1 took 28.072 ms
...
Training #50000 took 23.496 ms
Training took 1734.043 s or 28.9 min
--- Saving models ---
...
$>
```

```
$> python predict.py test.csv
Opened models.csv
--- Loading models ---
...
Opened test.csv
Created submission.csv
--- Testing ---
Testing #1 took 31.186 ms
...
Testing #50000 took 27.573 ms
Testing took 1786.428 s or 29.774 min
$>
```

The data for training and predicting has following structure:

- Each board has 20x20 cells, total 400 cells in one board.
- Training data has 50000 games, and you must predict another 50000 games from the test data.
- The first row represents the fields:
  `id` - id of the game
  `delta` - the number of steps between the start and stop boards (from 1 to 5)
  `start.1` - row 1, column 1 of the game's starting board
  `start.2` - row 2, column 1 of the game's starting board
  ...
  `start.400` - row 20, column 20 of the game's starting board
  `stop.1` - row 1, column 1 of the game's stopping board
  ...
  `stop.400` - row 20, column 20 of the game's stopping board
- Next 50000 rows represent boards of each game.

You can use Kaggle to test your program's accuracy.

Your models files and submission.csv will be checked during the correction.

You must be before Everything Dead Benchmark in the Kaggle's private leaderboard (score less than 0.14367).

Chapter VI

Bonus part

If you are absolutely certain that you completed all mandatory part, here are some ideas for bonus points:

1. Anything that you'll find truly remarkable about your algorithm.
2. Reach the top 42 in the Kaggle's private leaderboard (less than 0.12851). Show how awesome your algorithm is!
3. Be in the top 5 in the private leaderboard (less than 0.11171). If you reach this mark, you are truly the master of predictions! You must open a fortune teller club.

Chapter VII

Turn-in and peer-evaluation

Turn your work in using your `git` repository, as usual. Only work present on your repository will be graded in defense.