Crawl-O-Matic-O-Matic: Automated Optimization of AI

Eli Ramthun (ebramth15@earlham.edu)
Earlham College Department of Computer Science
Richmond, IN 47374

Introduction

Abstract
This project involves utilizing machine learning techniques to attempt to optimize an already existing artificial intelligence (AI) agent that plays the roguelike game Dungeon Crawl Stone Soup (DCSS).[1] Roguelike games feature high levels of randomization and are notoriously difficult - 0.73% of DCSS games played online are winning games.[2] An AI agent has been developed that is capable of winning a game of DCSS without human assistance. This project aims to utilize machine learning to automatically tune the agent to improve its performance for a certain portion of the game.

Background
AI's designed to beat roguelike games have existed almost as long as the games themselves - the "expert system" Rog-O-Matic was developed to beat the game *Rogue* (1980) as early as 1985.[3] Machine learning strategies have been effectively employed to optimize AI for games from chess[4] to other roguelikes such as Desktop Dungeons.[5]

This sort of expert system optimization has applications outside of automatic gameplay. In a 2005 work titled "Dynamic Asset Protection & Risk Management Abstraction Study", a team of researchers created a compelling case for the application of expert systems for purposes such as automated computer network security as a response to unpredictable threats, specifically naming Rog-O-Matic as an example system.[6]

Optimization Focus

This project seeks to optimize qw's **berserk logic** on the **first floor of the dungeon**. Figure 1 depicts the game.

Berserk Logic

- Berserking is essential for survival in some cases, but has serious - sometimes lethal - drawbacks
- qw's logic for berserking is numerical and well suited for computation:
  - if player.level < monster.scariness, then berserk

First Floor

- Small set of possible monsters to encounter (≈12)
- Completing the first floor of the dungeon takes only a few seconds, enabling generation of large amounts of data and faster validation
- Considered one of the more interesting and challenging segments of the game for human players

Figure 1. *Beginning of a game of DCSS. The player character is wearing green boots and wielding a sword; three exits are visible, and a fierce gnoll wielding a halberd threatens the player's progression further into the dungeon.*

Analysis

Statistical analysis of results was conducted with Chi-squared testing at a significance level equal to 0.05.

- The first SGD model was 1.01% worse than the default behavior with P = 0.0141.
- The second SGD model was 0.88% worse than the default behavior with P < 0.0001.
- The third SGD model is similar to the default behavior; it is 0.17% smaller, but with P = 0.2404, we cannot reject the null hypothesis.
- The LogisticRegressionCV model performed 0.08% higher than the default behavior, but P = 0.6186. Therefore, the model does not show a statistically significant difference in performance from the default behavior.

Parallelization & Data Collection

Figure 2 depicts the parallelized process for data collection. Table 1 depicts the data collected for training.

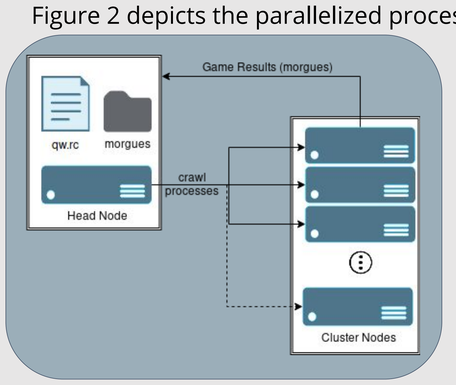

Figure 2. *This figure depicts the process by which data was collected. A head node sent out simultaneous instructions to run DCSS on the 12 compute nodes of the AI-Salam cluster. When a game finished the first floor of the dungeon, a morgue file was created and the results were sent to the head node.*

Node	Trials	Wins	Deaths	Scariness Emphasis	Scariness encoding	Rate of Success
as0	8902	8539	363	qw Default Behavior	0005000003	95.92%
as1	8871	8425	446	Adder	5000000000	94.97%
as2	8851	8261	590	Bat	0500000000	93.33%
as3	8870	8409	461	Giant Rat	0050000000	94.80%
as4	8895	8541	354	Gnoll	0005000000	96.02%
as5	8865	8377	488	Goblin	0000500000	94.50%
as6	8875	8418	457	Hobgoblin	0000050000	94.85%
as7	8867	8374	493	Jackal	0000005000	94.44%
as8	8897	8487	410	Kobold	0000000500	95.39%
as9	8876	8430	446	Leopard Gecko	0000000050	94.98%
as10	8875	8429	446	Worm	0000000005	94.97%
as11	8868	8358	510	PANIC! Always berserk	5555555555	94.25%
as12	8870	8410	460	Null - Never berserk	0000000000	94.81%
Total	**115382**				**Average**	**94.8648%**

Table 1. *This table depicts some of the collected data used to train the machine learning models. This table shows results for 115,382 runs of the first floor of DCSS, alongside the different berserk strategies tested. These (and other) results were aggregated and used to train machine learning models. A total of 260,977 data points were generated prior to training. All training runs were conducted using the race/class combination "Human Berserker."*

Discussion

- The attempted optimizations of qw's berserk logic failed to improve the success of human berserkers in traversing the first floor of the dungeon.
- Initial testing with a weaker character race, Deep Elves, shows promise that the LogisticRegressionCV model can indeed create significant performance improvements for the focused optimization.
- Deep Elf testing showed a 2.16% improvement in success rate with P = 0.0028, showing statistically significant improvement.
- Under default behavior, Deep Elves succeeded at a rate of 86.86%, much less than the Human rate of 96.05%.
- This indicates that such optimization methods may be more impactful when initial rates of success are lower.

Software Components

DCSS - open source turn-based game with procedurally generated content and permanent character death
qw - Expert AI hand coded to beat crawl with victory rate of approximately 17%[7]
Scikit-learn - open source, python based machine learning toolkit used computer optimum values for qw's parameters[8]
Bash - used to run crawl and aggregate data

Hypothesis

Utilizing machine learning techniques, a more optimal set of weights for the danger level of monsters in DCSS can be ascertained so as to improve the ability of qw to succeed in clearing the first floor of the dungeon.

Results

Machine learning models were generated using the collected data as input. The first three models were generated with Scikit-learn's Stochastic Gradient Descent (SGD) classifier, and another model was generated with Scikit-learn's Logistic Regression Cross Validation classifier. After creating models, every possible permutation of monster scariness weights was tested and the highest probability weight for each model was selected for trials. Table 2 contains the results of these trials.

Experiment											Trials	Success Rate
Default Settings	0	0	0	5	0	0	0	0	0	4	29,717	96.05%
SGD_1	4	0	4	4	0	0	4	4	0	4	2,474	95.04%
SGD_2	4	0	4	4	0	4	4	4	0	0	49,394	95.17%
SGD_3	0	0	0	4	4	0	0	0	0	0	49,451	95.88%
LogisticRegressionCV	0	0	0	4	0	0	0	4	0	4	28,510	96.13%

Table 2. *Since SGD generates models using some amount of randomness, multiple models created from the same data can vary. By tuning the hyperparameters of the model, even more variance can be introduced. Logistic Regression with Cross Validation will give a fixed result for given input data, and it performs cross validation of different hyperparameters to determine the most effective combination. In this table, monster scariness values are listed under the sprites of the given monster for a given experiment, with higher numbers and redder squares representing scarier monsters and increased likelihood of berserking. The results of default settings are included for comparison.*

Future Work

- Further tuning of hyperparameters could improve the results of SGD modeling.
- Creating models from data based on weaker character races shows promise based on initial Deep Elf analysis.
- This sort of parameter tuning could be performed for different parts of qw, optimizing progression through more of the game than the first floor.
- These techniques also have potential applications for software optimization in fields unrelated to gaming, such as search and rescue drone navigation.

References: [1] "Dungeon Crawl Stone Soup." [Online]. Available: https://crawl.develz.org/. [2] Colin, *Analyzing completed games ('morgue files') of Dungeon Crawl Stone Soup: colinmorris/crawl-coroner*. 2018. [3] A. K. Dewdney, "An expert system outperforms mere mortals as it conquers the feared dungeons of doom," 25-Jun-2016. [Online]. [4] T. Mitsuta and L. M. Schmitt, "Optimizing the Performance of GNU-chess with a Genetic Algorithm," in Proceedings of the 13th International Conference on Humans and Computers, Fukushima-ken, Japan, Japan, 2010, pp. 124–131. [5]V. Cerny and F. Dechterenko, "Rogue-Like Games as a Playground for Artificial Intelligence – Evolutionary Approach," in Entertainment Computing - ICEC 2015, 2015, pp. 261–271. [6] G. Henderson, E. Bacic, and M. Froh, "Dynamic Asset Protection & Risk Management Abstraction Study," p. 50, 2005. [7] G. Henderson, E. Bacic, and M. Froh, "Dynamic Asset Protection & Risk Management Abstraction Study," p. 50, 2005. [8] elliptic, The DCSS-playing bot qw. 2018.

Earlham COLLEGE

Acknowledgements
Thanks to Dr. David Barbella for his motivational and informative advising, Dr. Xunfei Jiang for her constant guidance and leadership, and Dr. Jose-Ignacio Pareja for his inspiration and encouragement. Additional thanks to all of my peers in the computer science department and to the DCSS community, including the ##crawl and ##crawl-dev IRC channels. Special thanks to Sigmund for staying out of the first floor of the dungeon.