Summary - How Can Genetic Algorithms Affect AI Models

Topic and Hypothesis

We implemented a genetic algorithm to train an AI to play a game, and compared the genetic algorithm method to traditional training. Our experiment was to see if the genetic algorithm method produces better results than normal training.

Connect Four has a 6 vertical by 7 horizontal board where players take turns dropping different colour pieces. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of your own colour pieces.

We used a game to judge the effectiveness of AI training because games have a clear winner and loser, and have rules to play by. Connect Four is relatively simple with only a few possible moves to choose from, but it is still more complex than a game like tic-tac-toe.

We hypothesize that genetic algorithms implemented into the Connect 4 AI model will perform slightly better than the non-genetic algorithms model. We think that the genetic algorithms model will win more games against a normally-trained model.

Background

Neural networks

Neural networks are a simplified computer model of neurons in the brain. A neural network processes an input through several layers of neurons and emits an output. The processing is controlled by numeric weights on each neuron, which control how the neuron activates other neurons in the network.

The neural network can recognize patterns in the input through the connections in the neurons. AIs for games usually use neural networks because humans also look for patterns in the shape of a game board to decide what moves to make.

Reinforcement Learning

For a game like Connect Four, a neural network can learn how to play well if it is taught which moves are good and which are bad. Teaching the network uses a process called Reinforcement Learning, where we modify the weights in the network.

When we know a move is good, we update the weights so that this move is more likely to be played in the future if the inputs are similar. When we know a move is bad, we update the weights so that this move is less likely to be played in the future. This means that good moves get rewarded over time and bad moves get penalized.

Genetic algorithms

Genetic Algorithms are a method to optimize a solution by randomly modifying parameters. Genetic algorithms were inspired and modeled around natural selection, where genes in living things randomly mutate over time, and gradually the living things adapt to fit their environment as the individuals with the best fit tend to reproduce more.

We can think of a set of parameters as “genes”. Random modifications to the parameters can be “mutations”. Parameters with the best fit are mutated to produce new parameters, while parameters that do not fit are discarded. Like with natural selection, we expect the population of parameters to gradually adapt to fit better.

Genetic Algorithms For Neural Networks

For neural networks, the weights on the neurons are the “genes”. We can produce mutations by randomly modifying the weights, and a model is more “fit” if it wins more games versus other models.

To implement a genetic algorithm, this means we need to maintain a population of models and have them compete against one another.

Procedure

We implemented the rules of Connect Four in a Python program, and used the pytorch library to build neural network models to choose moves to play in a game.

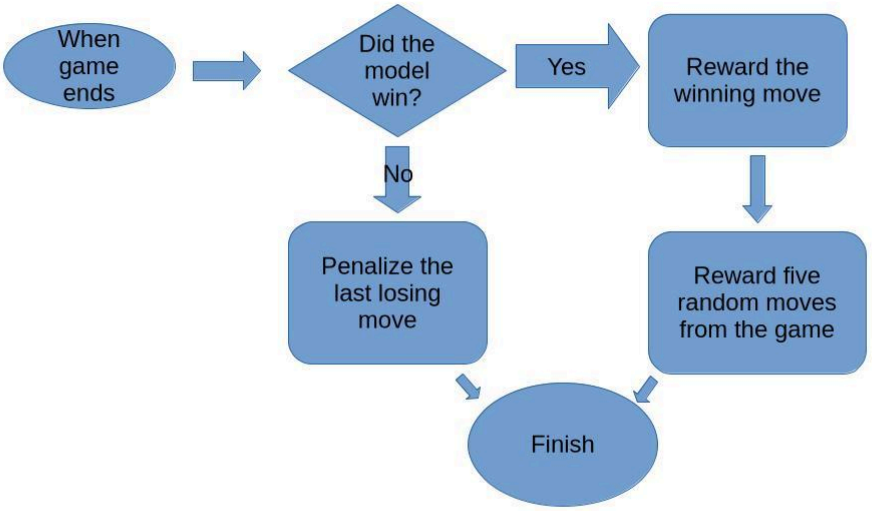
Comparison method

To judge whether one model is stronger than another, we have them play 2000 games against one another. We consider the model that wins more games to be stronger. We check the strongest normal and genetic models against one another periodically.

Reinforcement Learning method

We train models by playing a series of 100 Connect Four games between one model and another. Humans are not involved in training or supervising the process.

After each game, we consider the last winning move to be “good” because it won the game. The loser’s last move is considered “bad” because it did not prevent the loss. On the winning side, we also choose 5 random moves from the list of moves the model played and consider those “good” as well, because they should have led to the winning situation.

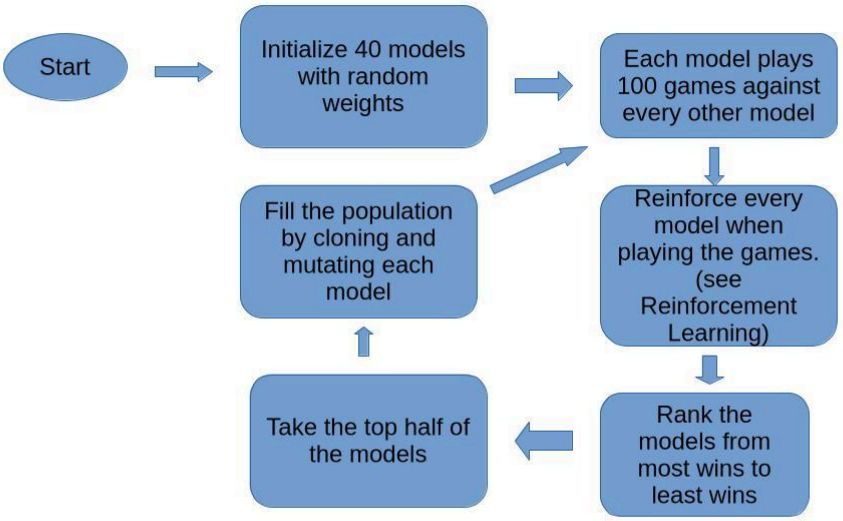


Genetic Algorithm Process

Our genetic algorithm maintains 40 models at a time, and has every model play 100 games against every other model.

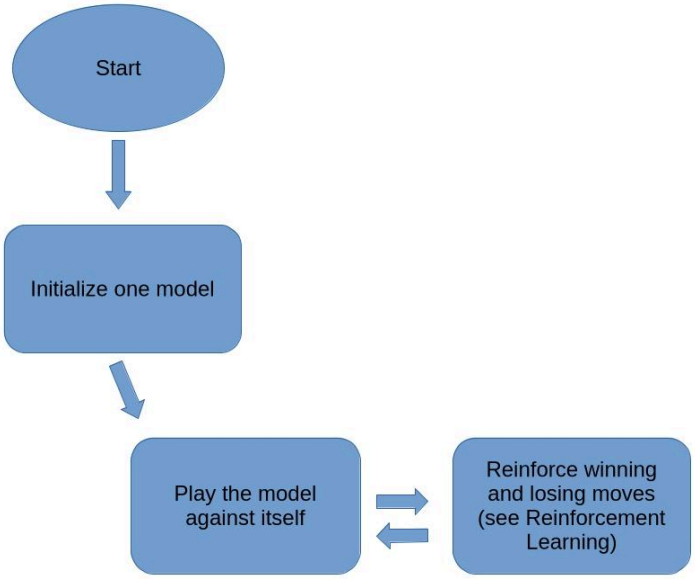
We rank the models by the number of games won, and remove the bottom half. Then we fill the population back to 40 by copying the surviving models and mutating random weights to produce new models. This mimics natural selection, where the highest performing individuals are able to pass on their “genes” to offspring. The offspring maintain many of the genes of their parents, but with some random changes.

This process is repeated over many generations to try to select better models.

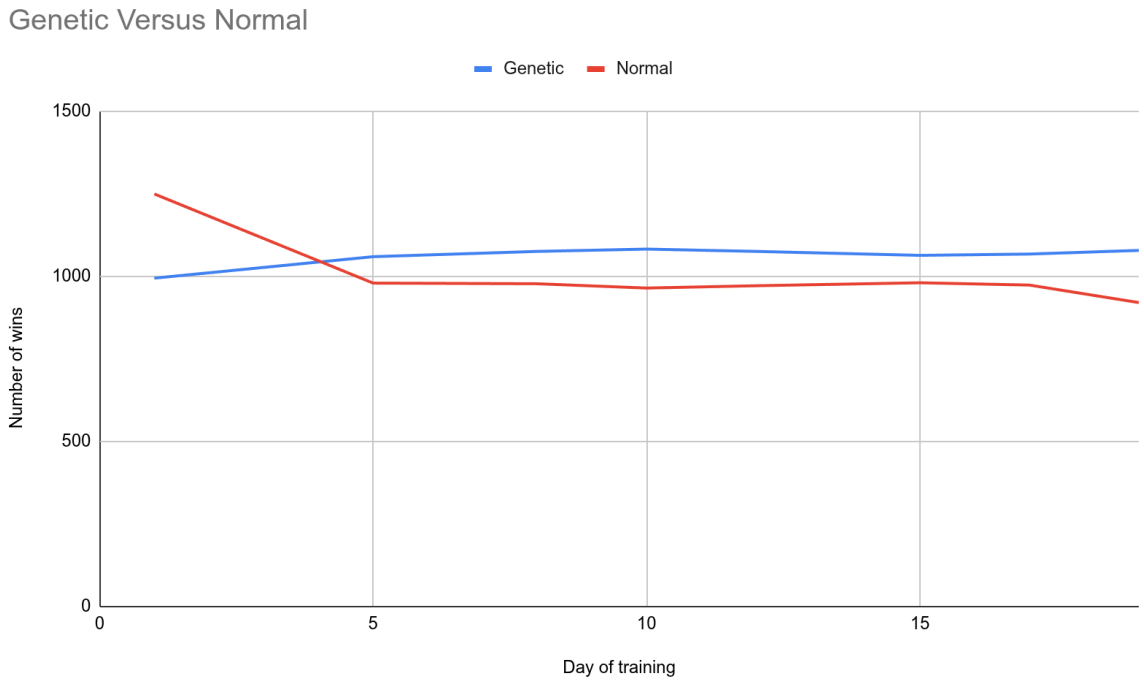


Normal Training Process

Normal training only maintains one model, and continuously plays it against itself.



Results and Conclusion



By the end of training, the genetic model won slightly more games over the normal model. This result validates our hypothesis to some extent.

However, by observing the moves that the models played, neither one seemed to understand how to win the game. They mostly played in the same column, not really reacting to an opponent's moves.

Despite that, the genetic model still performed slightly better than the normal model. We do not fully understand why this is the case, since both models always seemed to play in the same places.

As a result, we are not very convinced that our hypothesis is actually true. It is hard to conclude that genetic algorithms are definitely better than normal training, but more investigation is justified.

Future Work

Efficiency of training

If genetic algorithms are able to achieve better results in the same amount of training time, then they could help reduce the amount of resources needed to get a result. This would mean we can save electricity and time developing AI.

Genetic algorithms work when fitness of models can be easily measured. When the fitness can be measured easily, this technique could be applied.

Limited Resources

We did not observe much improvement in the play of either normal or genetic models. It's possible that if we used more time and compute power that we would have achieved better results.

Other questions

- Could we apply this technique to other games?
- What other applications are there for this technique besides games?

Sources of Error

- Coding errors
- Limited time and compute power
- Better choices of genetic and neural network parameters