Comparison of Traditional Machine Learning Methods against the Literature Implementation of a NN-Genetic Algorithm for Playing Classical Video Games

Nícolas Bohm Agostini, Matan Kaminski, Sabin Park, Lichen Wang

EECE 5644 – Machine Learning and Pattern Recognition

Dr. Jennifer Dy and Dr. Deniz Erdogmus

November 17, 2016

Northeastern University
Department of Electrical and Computer Engineering

Introduction and Project Idea:
For years, computer scientists and engineers have dreamt of the day a full system framework can master a wide range of diverse tasks by itself. With the recent advancements in machine learning and artificial intelligence technology, achieving this goal is not so far off in the future. However, before creating such systems for our complex day-to-day tasks, research has been focusing on developing a system framework that accomplishes a relatively simple activity many of us have spent hours upon hours doing during our childhood: playing video games.

In this project, we will find out how modern machines and various machine learning algorithms fare against classical video games such as Pacman and Super Mario World. Instead of dedicating hours upon hours to play these video games with less than perfect results, we aim to solve this problem by programming a computer to beat these games for us. This project provides an engaging hands-on experience in developing machine learning models that utilize game frames as inputs to make fast-pace decisions, starting with little domain knowledge on how the game mechanics work.

We will compare different decision-making algorithms in an attempt to create a better video game player. We will try both supervised and unsupervised machine learning algorithms and evaluate their performance.

Current Methods:
There are various examples where machine learning principles are applied to video games. In one such example, Google Deepmind partnered with Blizzard to attempt reinforcement learning on Starcraft 2 games [1]. This is not the first time that Google's Deepmind has been used to learn how to play video games. In fact, Deepmind has already been taught to play 49 games [2].

In [5], artificial neural networks with several different fitness functions were used to implement self-playing games of Atari's classical video game, Pong. Results provided unreliable control inputs for the controller trained by the neural networks. As such, one of the purposes of our research project would be to explore how other traditional machine learning methods may improve upon existing video game AI.

The *References* section has additional information on previous research performed in this field [3][4][5].

Approach:
Stanley has already implemented a genetic algorithm that was able to learn how to play Super Mario World [6]. However, genetic algorithms rely heavily on chance to provide acceptable results, and also require a great number of iterations. Some argue that genetic algorithms are not in the same category as accepted traditional methods for machine learning. As such, Stanley's genetic algorithm method is not necessarily the most efficient solution to realizing a successful video game player. Thus, our goal is to modify his algorithm and add extended functionality to our game of choice. To accomplish this goal, we will take the following steps:

1. Decide on a scoring function that uses a common value to compare the performance of different algorithms.
    a. Depending on the final game state, the score will be a function of:
        i. Player distance from start position.
        ii. Player survival time.
        iii. Player's score in the game.
    b. Training time. If we train a model under same time or optimization loop, by compare which model perform better, we can demonstrate which model is effective and efficient in the training process.

2. Input data and features:
    a. We will look at a frame of the game as the core input data/features we will feed our algorithms.
    b. We will extract the player position, enemy position(s), walls, and platforms from each frame as the features input to the

3. Implementing the algorithm: At a minimum, we aim to implement one unsupervised and one supervised learning algorithm. Ideally, we want to explore how the various algorithms will perform, and identify their strengths and weaknesses. See *Table 1* below for some of the directions we want to take this project. Since this proposal explores multiple ideas, we will start with one algorithm and see which ones we want to implement based off of the results.
    a. For supervised learning, we will randomly sample different game configurations from a human player run; then feed these "samples" to the learning model in order to train it; and later, let the algorithm play the game to generate a score.
    b. For unsupervised learning, we will allow the algorithm to run over multiple iterations of the game, comparing score functions at the end of each run.

Table 1: Summary of algorithms

Supervised	**Unsupervised**
Neural network with backpropagation	Genetic algorithms applied to NN parameters with random initialization (implemented on paper)
Bayesian decision	Genetic algorithms applied to NN with pre-trained parameters (from the best runs of our supervised implementation)
Feature extraction combined with a different supervised learning algorithm ● Distance Feature ● Clustering on Frame ● Handcraft Label	Genetic algorithms applied to NN parameters using Bayesian parameter estimation to decide upon the parameters to next breed
	Neural Net with gradient descent optimizations for maximizing score

4. Compare the final scores as well as the training time of the various algorithms to see which one is better. Analysis the performance of each method.

Expected findings from this Investigation:

Comparing the results to the Stanley's paper [6], we expect that:

- The pre-initialized genetic NN will train faster than the randomly initialized genetic NN
    - It is not guaranteed that the pre-trained neural network will reach a better score;
- The supervised learning algorithms will train faster than the unsupervised algorithms given its relative simplicity and the different approach;
- The unsupervised learning algorithm will be able to achieve a better final score, whereas the supervised learning algorithm will be able to achieve similar performance to our training data.

Overall the progress we hope to make is faster training with better results as compared to just the simple genetic neural network from [6].

Expected Results Affecting the Research Community:

We hope to draw some conclusions from our project in relation to decision-making given world states. By examining both the outcomes in performance and the training time of various algorithms in this controlled, small-scale environment, we may be able to project what results can be achieved on a larger scale scenario. This will then provide insight into knowing what type of training algorithms are suitable for complex games, or perhaps other, more practical, tasks.

References:

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

[2] Deep Mind. (n.d.). DeepMind and Blizzard to release StarCraft II as an AI research environment | DeepMind. Retrieved November 17, 2016, from https://deepmind.com/blog/deepmind-and-blizzard-release-starcraft-ii-ai-research-environment/

[3] Thrun, S. (1995). Learning To Play the Game of Chess. *Advances in Neural Information Processing Systems*, *7*.

[4] Wong, K. W. (2007). Adaptive Computer Game System Using Artificial Neural Networks. In *Neural Information Processing* (pp. 675–682). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-540-69162-4_70 Accessed November 17.

[5] Ramirez, L. E. (2014). Using Artificial Neural Networks to Play Pong.

[6] Stanley, Kenneth O, and Risto Miikkulainen. 2016. "The MIT Press Journals Evolving Neural Networks through Augmenting Topologies." Accessed November 17.

Additional References:

Huang, B.-Q., Cao, G.-Y., & Guo, M. (2005). Reinforcement Learning Neural Network to the problem of Autonomous Mobile Robot Obstacle Avoidance. *International Conference on Machine Learning and Cybernetics*, *1*, 18–21.

Penfol, H. B., Diessel, O. F., & Bentink, M. W. (1990). A Genetic Breeding Algorithm Which Exhibits Self-Organizing in Neural Networks. *International Symposium on A.I. Applications and Neural Networks*, 293 – 296. Retrieved from http://www.cse.unsw.edu.au/~odiessel/papers/ainn90penfold.pdf

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (n.d.). Playing Atari with Deep Reinforcement Learning. *arXiv Preprint*, *arXiv:1312*.

Beasley, J. ., & Chu, P. . (1996). A genetic algorithm for the set covering problem. *European Journal of Operational Research*, *94*(2), 392–404. http://doi.org/10.1016/0377-2217(95)00159-X

Ben-Tal, A., & Nemirovski, A. (1998). Robust Convex Optimization. *Mathematics of Operations Research*, *23*(4), 769–805. http://doi.org/10.1287/moor.23.4.769 Accessed November 17.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484-489. http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html Accessed November 17.