Evolutionary Learning In Two Stage Gene Networks

Josh Pattman

April 30, 2024

1 Introduction

Two of the most important features when describing an organism are its genotype and its phenotype. A genotype encompasses the entire genetic code of an organism, while the phenotype is the physical manifestation of that code as specific traits such as size or color. The relationship between a genotype and a phenotype is a complex one, as a single locus in the genotype does not hold a one-to-one relationship with a single trait in the phenotype. Instead, the phenotype is the result of the entire genotype interacting with the gene regulation network (GRN) of the organism.

A GRN is a mechanism by which the various genes in the genotype interact by stimulating or repressing each other, eventually resulting in a stable state describing the phenotype. This is an extremely complex process, so for this research, it is simplified to a directed graph. Each node in the GRN graph represents a locus in both the genotype and phenotype, and connections between nodes represent the interactions between genes. The topology of the GRN is not fixed throughout generations, but instead evolves alongside the genotypes of the organisms, with each organism having its own unique GRN. However, the rate of evolution of the GRN is significantly slower than that of the genotype, meaning that the GRN may retain information from previous fitness landscapes, even if that information is not immidiately useful to a given organism at the present time.

Evolved GRNs have a number of interesting properties. For example, a GRN may recognise genes which frequently covary, and increase the likelihood of those genes being expressed together. A GRN also may take advantage of its ability to retain information to increase the likelihood of a group of traits from a previous landscape being expressed in the future. This is particularly useful in nature, as a previously proven trait of a creature may be more helpful in a future environment than a new randomly evolved one. This ability of a GRN to remember previous fitness landscapes will be reffered to as genetic memory throughout this research.

This paper aims to investigate the effects of genetic memory in a variety of scenarios, building upon previous work by Richard A. Watson Et al. CITEME, henceforth referered to as the previous work. In the previous work, a specific mathematical model for a genotype and GRN was created, and a hill climbing algorithm was performed to optimise a GRN to commit multiple phenotypes to memory. The trained GRN was then used to produce various combinations of the training phenotypes upon being presented with a random genotypic input.

The previous work also showed that, using their specific method, the weights of the GRN tended to move towards the weights acheived by Hebbian learning on the training data. This result is significant, as it suggests a hill climbing algorithm may in part be equivalent to more traditional 'intelligent' machine learning algorithms.

This paper aims to initially reimplement the algorithms of the previous work, and reproduce the results obtained. Following this, an extension to the previous work is proposed, which aims to more accurately model the interactions between genes in a GRN. This extension is then extensively tested and the results are compared to the original work.

2 Experiment Setup

To represent the genotype of an organism, a vector G is used. This vector has an element per gene in the organism, which in many of the presented cases represents a pixel in an image. Each element can be any in the range $[-1, 1]$.

The GRN is represented by an interaction matrix B, which has the same number of rows and columns as there are elements in G. The state of the GRN at time t is represented by a vector $P(t)$, which is governed by (1) and (2). $P(t)$ has no upper and lower bound.

$$P(1) = G \tag{1}$$

$$P(t+1) = P(t) + \tau_1 \sigma(BP(t)) - \tau_2 P(t) \tag{2}$$

Where τ_1 and τ_2 are the update rate and decay rate of the network respectively, and σ is the $tanh$ function. The final phenotype on which the fitness is evaluated is given by $P(t_{final})$, where t_{final} is set to 10 for all experiments in this paper.

The fitness function is a modified version of that used in the previous work and is given by (3).

$$|sum(P(t_{final}) \odot T)| \tag{3}$$

In this equation, T is the target phenotype and $\odot$ is the element-wise multiplication operator. Using this function, a value of 0 is minimally fit, but the fitness can extend to positive infinity. The reason for taking the absolute is that a perfect negative of the produced phenotype is indistinguishable from the target phenotype for the GRN. In the previous work, the fitness function would penalise a perfect negative, however this paper chooses to reward it.

The genetic algorithm (GA) used in this paper is a simple point mutation hill climbing model. At any given time step, there exist two individuals. One of these represents the previous best individual, and the other is a mutated version of the previous best. At each step, the mutated individual is re-initialised to the previous best, then a random point mutation is applied to both its genotype and GRN. If the mutated individual has a higher fitness than the previous best, it becomes the new previous best. This process is repeated for a set number of iterations, and the final best individual is returned. It is imperative that the point mutation size of the GRN is significantly smaller than that of the genotype, otherwise the GRN may overwrite previously learned information. Mutations are either a uniformly or normally distributed random number, which is added to the previous value of the gene.

3 Reimplemented Results 1pg

The first experiment that was reimplemented was experiment 2, shown in Figure 2 in the original work. This experiment aimed to show that the GRN was capable of remembering multiple phenotypes, and was able to reproduce them in future iterations. It also aimed to perform an in depth analysis of the behaviour of the state of the GRN though the timesteps.

For this experiment, SPEAK ABOUT THE EXPERIMENT PARAMETERS.

Figure 1 shows how the values of each of the various values of the GRN matrix change thoughout the generations. Each value is represented by a line, with the x-axis representing the generation number and the y-axis representing the value of the matrix. It is clear to see that there are three distinct groups of parameters: increasing, decreasing, and static. The increasing and decreaseing groups are due to the GRN learning either a positive or negative correlation between the genes that cell connects, whereas the static group is due to the GRN learning that the genes are not correlated.

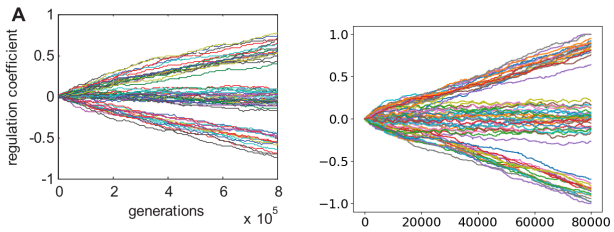

Figure 1: Values in B over generations. On the left, the original paper. On the right, the reimplemented results.

Figure 2 shows a visual representation of the matrix B at the end of training. An identical

pattern to that of the original paper emerged, however the weights have a different scale. This is expected, and is due to small changes in the implementation, such as mutation rate. Similarly, Figure 3 shows the weights obtained using Hebbian learning for B. As expected, the weights are very similar to those obtained through hill climbing, suggesting that the hill climbing algorithm may approximate Hebbian learning in this situation.

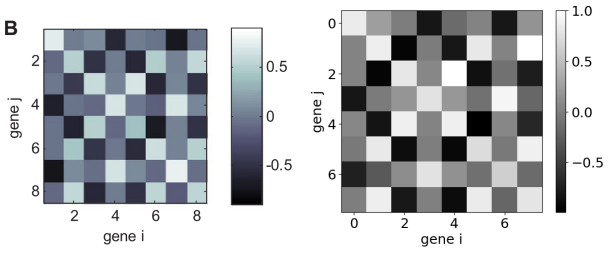

Figure 2: The final weights in B obtained through a hill climbing process. On the left, the original paper. On the right, the reimplemented results.

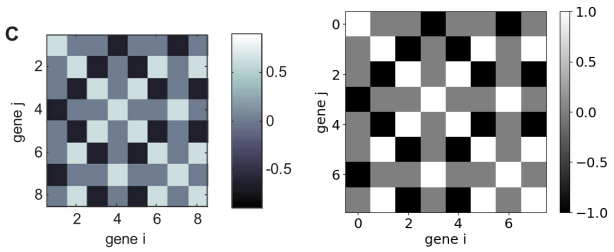

Figure 3: The weights obtained using Hebbian learning for B. On the left, the original paper. On the right, the reimplemented results.

Figure 4 demonstrates how the values of P change over the 10 developmental timesteps. Each gene in P is represented by a line, with the x-axis representing the timestep and the y-axis representing the value of the gene. The results presented by this paper clearly follow the same pattern as those from the original work, with the genes all converging to either a stable positive or negative state by the end of the 10 timesteps. There is some correlation between a genes starting value and the state it finally ends up in, but the figure shows that this is not the only factor.

Figure 5 shows a selection of phenotypes produced by the GRN when given a random genotype as input. The results are very similar

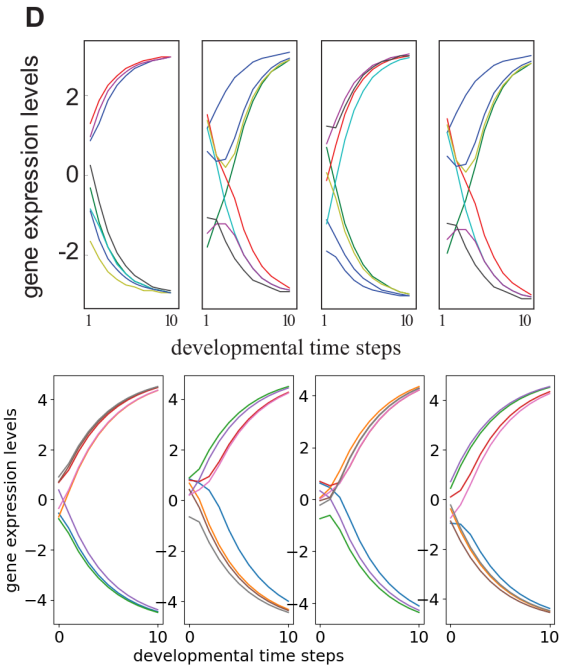

Figure 4: On the top, the original paper. On the bottom, the reimplemented results.

to those of the original paper, with the GRN producing both targets that are present in the training data. The GRN also produces the inverses of the two targets, which is to be expected.

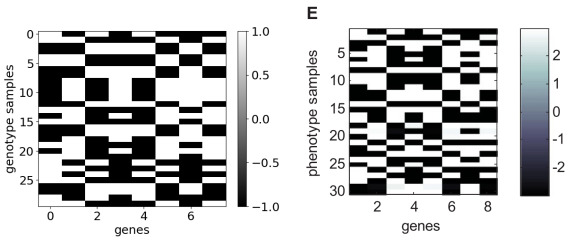

Figure 5: On the left, the original paper. On the right, the reimplemented results.

4 Extension 1pg

The previously described model of a GRN is a significant abstraction of the true complexity of the interactions between genes. To extend this work, this paper proposes a more advanced model of a GRN which takes inspiration from protein production in genes.

In a biological GRN, the expression level of a gene is affected not directly by the expression levels of other genes, but instead by the

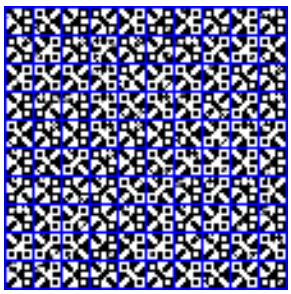

Figure 6: Bestiary of genotypes reimplemented to the specification of Figure 4 in the original paper.

levels of protiens produced by those genes. It also may be the case that there are significantly fewer unique protiens available for this purpose than there are genes, meaning that genes with similar functions may have to evolve to use similar proteins for regulation. These features of GRNs are not caputered by using a single regulation matrix to represent the interaction between genes.

4.1 Goals and Hypothesis

This extension focusses on simulating these intermediate messenger proteins in a simplified manner while maintaining maximum similarity with the previously described method, enabling simple comparisons between the two.

The hypothesis of this experiment is twofold:

1. **The gene regulation network will be able to remember previously expressed phenotipic patterns even with an overall reduction in the number of parameters when compared to a dense network.** This is because in the datasets used for these experiments, and indeed real life, there are patterns that the network should be ableto encode into a single protien. Each pattern may then also affect multiple genes in a similar way. These features leave much room for the network to genralise and group certain features that behave in similar ways together.

2. **The gene regulation network may become more robust and better at genralising to new patterns, as it is forced to 'make sense' of the input genotype.** In traditional machine learning, a bottleneck is a point in a model where there are fewer features than there were inputs to the model, forcing the network to learn the patterns in its data rather than memerising it. Hypothetically, this should also apply to a GRN with fewer chemicals than genes, as the GRN will need to extract only the most important information from the inputs in order to best create an accurate output.

4.2 Model

To represent this new model of a GRN, the singe interaction matrix B must be replaced by two distinct matricies, one for encoding a state of the GRN into a vector representing the protein production, B_e, and one for decoding a protien production vector into the updates to each state, B_d. Given a number of available proteins $n_{protein}$ and the genotype size $n_{genotype}$, B_e has dimensions $(n_{protein}, n_{genotype})$, and B_d has dimensions $(n_{genotype}, n_{protein})$. Given these matricies the phenotype at the next timestep, column vector $P(t+1)$, is given by the equation

$$P(t+1) = P(t) + \tau_1 \sigma(B_d(B_e P(t))) - \tau_2 P(t) \tag{4}$$

Where τ_1 and τ_2 are the update rate and decay rate of the network respectively, and σ is the sigmoid function.

Mutations also work in a different way in this model. Instead of directly mutating the values of B, the values of B_e and B_d are mutated. Mutations are performed on B_d in the same way as they are applied to B in the original model, however, mutations on B_e can only mutate values of the matrix to 1, 0, or -1.

The reason behind this is primarily to allow the evolutionary process to easily invert and enable or disable the effects of one gene on another. The prescision of mutating a floating point value is not needed here because finer control can be performed by the second matrix, B_d. Combining these two effects together allows the evolutionary process to make both more drastic and also more subtle changes to the GRN, which hypothetically may allow it to more easily recover from local optima.

This simplification of B_e also significantly reduces the parameter space of the model, which should benefit rate of learning. However, due to the fact that changes to B_e have a much greater effect on the output than changes to B_d, the chance of mutation on B_e must be much lower than that of B_d. In these tests, the chance that B_e was mutated instead of B_d was 0.005.

This method has some useful properties aside from furfilling the aim to simulate the intermediate properties:

1. Mapping the previous state to the next state with two matricies may be a faster computation with fewer parameters compared to a single interaction matrix, as long as

$$n_{protien} < n_{genotype} \div 2 \tag{5}$$

2. As the matricies are multiplied linearly together, it is possible to find the equivalent single interaction matrix by performing the calculation

$$B = B_d B_e \tag{6}$$

Computing the single interaction matrix enables simple comparison to both the original approach, and the weights derived from Hebbs rule, which is useful when evaluating if this model maintains the tendancy to move towards the Hebbian weights.

- TODO: Support the value of asking that question (using literature)

5 Extension Results

- 1.5 page

Compare bestiary of dense to bitdense
Compare different number of proteins
Compare number of parameters
Compare training speed

6 Conclusion

This study has not only shown the reproducibility of the results obtained by R. Watson et al [**?**], but has also contributed some valuable insights into improvements which can be made upon the model of a GRN.

- This paper has presented results which agree with the original conclusion that the hill climber tends towards the hebbian weights.

- However, upon implementation of the extension, it was shown that similar results could be obtained even with significanty different weights to the hebbian ones.

- This may suggest that the hill climbing algorithm is not as similar to hebbian learning as previously thought, or that the extension model is not as effective as the original model.

- The extension model was also shown to be faster training and more robust than the fully dense network.

- A major critique of this method is that the weights of -1, 0, and 1 are straying quite far from real-life biology, where each gene may release a concentration of a certain protein.

- However, the aim of these experiemnts was never to make a perfect biological model, but instead a rough approximation which could be used to explore the effects of genetic memory in a GRN.

- Bitdense is also very sensitive to learnging rate, bit probability, and generations before switch, making it harder to tune and get right

- Running some sort of meta hill climber than runs multiple bit dense sims at once, each with own learning rate, and has competition between different sims may be future work, allowing the optimal params to be found.

- It is also very possible to come up with better but less biologically training algorithms, which may be interesting to explore in the future (some sort of GAN-inspired architechture / competitive co-evolution)

7 Appendix