Lindsey Sample

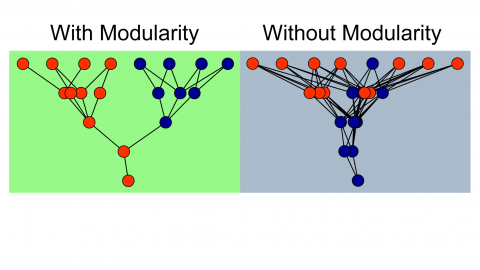
COGS 319

Introduction

**Introduction**

What is the optimal architecture for neuro-controllers? How do brain structures evolve with respect to the behaviors that they support? What are the tradeoffs between efficiency and full access to input information? These are central questions in both Cognitive Science and Neuroscience, but it is difficult to address them given the complexity of animals. We can use models to examine hypotheses regarding brain architecture and the effect on fitness that variations can have. Presently, we are interested in examining the differences in neural architectures that evolve in dynamic versus static environments in evolved artificial neural networks (ANNs). Eventually, we hope to expand this project to include a comparison of ANNs with Weight Agnostic Neural Networks (WANNs), as well.

Often, Cognitive Scientists have used neural networks to model the brain of biological organisms. The standard neural net features a suitable architecture of nodes, and requires the development of appropriate weights between nodes that yield optimal output or performance. When exploring architectures of neurocontroller, a common method of describing and analyzing architecture is to examine modularity and sparsity. This attempts to quantify the degree to which the system is divided into modules, or is more fully-connected. Clune, Mouret, & Lipson (2013), define modular networks as those which “contain highly connected clusters of nodes that are sparsely connected to nodes in other clusters.” Another way to conceptualize modularity is to examine how discrete or integrated communication is within the network between input (sensors) and output (motors).



**Figure 1.** Depiction of ANNs with modularity and without modularity(Huizinga, Mouret, & Clune, 2014).

Modularity is believed to evolve from selection pressure **to reduce connection costs** (Clune et al., 2013). It is important to note the distinction that modularity does not evolve from selection pressure on performance alone, but specifically on pressure to reduce connection costs (Clune et al., 2013; Livingston et al., 2016).

The modularity of nervous systems is hypothesized to enhance evolutionary adaptation of a population by allowing selection to target regions separately. Previous research suggests modular networks are favorable for handling evolutionary goals. Livingston et al. (2016), suggests modularity can allow rapid response to environmental change, specialization without loss of useful sub-functions, and avoidance of “catastrophic forgetting.”

Kashtan and Alon (2005) found that the evolutionary emergence of modularity was related to the environment faced and evolutionary goal experienced. They found that when goals were repeatedly switched (with common subgoals), the networks rapidly evolved to satisfy the different goals with only a few rewiring changes. They claim that these evolutionary forces favor modularity for its structural simplicity and ability to rapidly adapt (Kashtan and Alton, 2005). They found that when the varying goals contained no common sub-goals, modular structures did not evolve and that adaptation was very slow (since evolution was essentially starting from scratch each time there was a change).

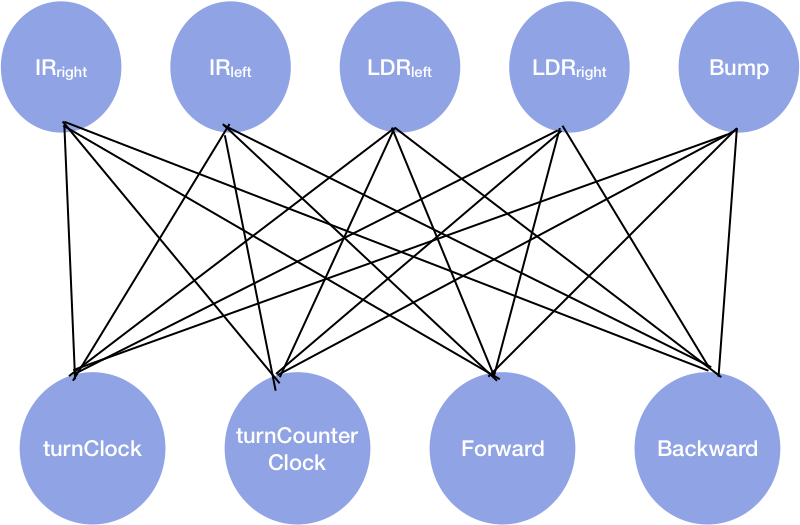
The majority of research on this topic points to the claim that cognitive architectures develop with respect to their specific environments, or the tasks required of them. Modularity evolves with respect to pressure to reduce connection costs in environments, and is a strategy for easily evolving to changing environments. This leads us to the hypothesis that modularity will be favored in dynamic environments, more-so than in static environments. We have developed a model to test this hypothesis.

We will test this hypothesis with a simulated robot called GridBot. GridBot‘s task is to traverse a GridWorld that features a light source and obstacles, which vary, in order to change what is expected of GridBot in the changing environment condition, and are stagnant in the stable condition. GridBot features simulated versions of two photoresistor sensors (LDRs), two infrared sensors (IRs), and one bumper, and navigates GridWorld autonomously, making movement decisions based on the values inputted and computed in its ANN. We will analyze and compare the best performing emergent architectures in the static and dynamic condition.

**Methods**

*GridBot and GridWorld.* Presently, we develop a simulated world to emulate a toy world for a terrestrial bot to traverse. This bot, called GridBot, has two light dependent resistors (LDRs), two infared sensors (IRs), and one bumper on its front. It can traverse a grid world by moving forward or backward, or by turning to the left or right. Each step, GridBot can make one move: stepping, or turning. GridBot can make 100 moves in a trial, and move backwards, move forwards, turn clockwise, or turn counterclockwise on any given move. Each move is determined probabilistically based on its ANN, featuring 5 input nodes corresponding to its 5 sensors, and 4 output nodes, corresponding to its 4 move options.

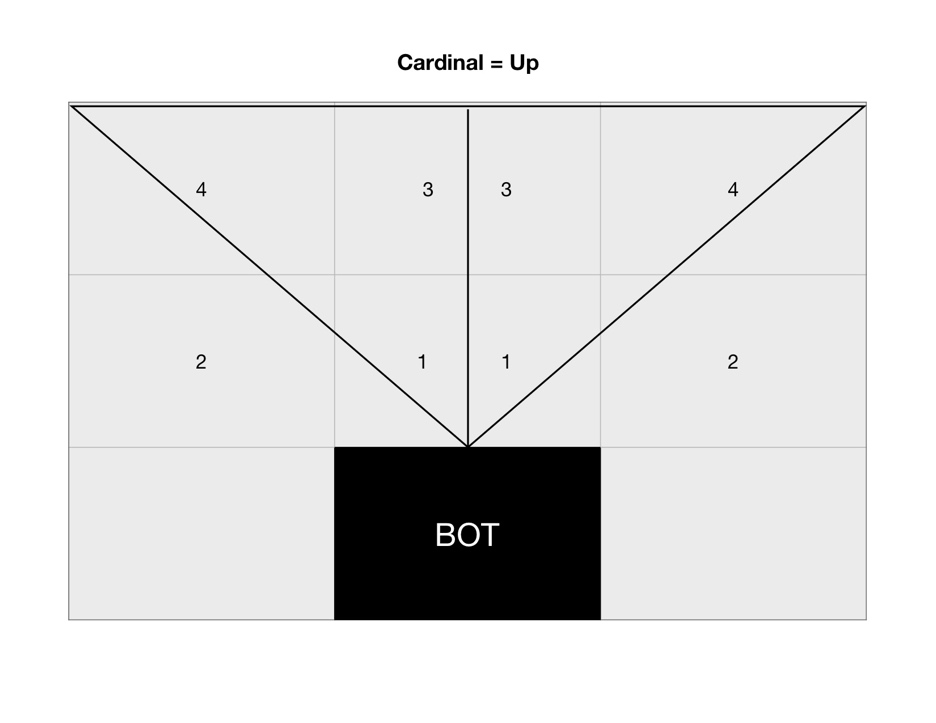
GridWorld is a grid, composed of a 10x10 (x,y) matrix (100 squares) in the stable condition, and a randomly determined NxN matrix (also equal to 100 squares) in the dynamic condition (4x25, 25x4, 10x10, 20x5, or 5x20). Every square within the grid contains a value of light that can be collected by entering that square, and a binary value that indicates if there is an obstacle in that square. In the static condition, there are 5 obstacles that form a square in the center of the GridWorld. In the dynamic condition, there are 5 randomly placed obstacle squares. GridBot cannot step into a square that an obstacle inhabits, and cannot collect light from these squares. Light values are highest near the source, and fall off in a gradient (255\*e^(-distance\*.5) across the GridWorld. In the static condition, the light source resides at the top of the world (5,5). In the dynamic condition, the light source is in the center top as well, but this depends on which dynamic grid the bot is randomly in (see **Figure X)**.



**Figure X.** Depiction of GridBot’s ANN

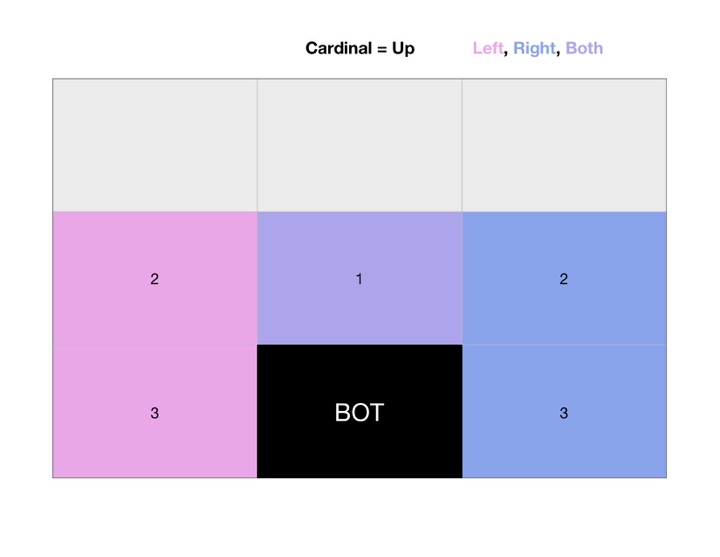
*ANN.* Each GridBot has a weight matrix, where weights can take on values of -1, -.5, 0, .5, and 1. On each turn, each output receives a value that is the dot product of the weight matrix (or the bot’s genome), and the input. The outputs are also scaled using a sigmoid function (1 / (1 + x^(-sum(input\*weight[,i])) and divided by the sum of outputs (to appropriately scale them for probabilistically choosing between them), and an output node is probabilistically chosen. This action is then taken, sensors are updated, and the process repeats for 100 moves. At the end of a trial, a GridBot has the amount of light it has collected accumulated.

*Sensors and Action.* IRs are calculated by searching the four spaces in front of the bot as depicted in **Figure X**. The spaces are searched in order, and if an object is detected in the space, the IR value becomes the distance between the bot’s current location in the grid and the obstacle.



**Figure X**. Specifications for LDR sensors (right and left).

LDRs are calculated by taking the average of the space the bot is currently in and the 3 adjacent spaces (depicted in **Figure X**). Each LDR assumes the light value of the space it is sensing according to the specifications described.



**Figure X.** Specifications for LDR sensors.

GridBot also features a bumper, that returns 1 when there is an object directly in front of where GridBot is facing, and 0 when the space is unoccupied.

*Selection.* GridBot survives by harvesting light. Each move, GridBot’s light counter is updated by adding the light it has collected by moving into a new square, and taking the mean of the light sensed in the square it is in, averaged with the 3 adjacent squares to the right (for the right sensor) and left (for the left sensor). The total light collected over a trial (X moves) will serve as that bot’s fitness, where the most fit bots collect the most light.

The evolutionary algorithm uses a roulette style wheel to probalistically select the next generation of GridBots based on relative fitness of the previous generation. The corresponding bot numbers are sampled based on their relative fitness, and each bot’s weight matrix is mutated at a rate of 5%. This means 1/20 weights are changed at random (from the list of possible values: -1, -.5, 0, .5, and 1).

The experiment is run for 100 generations, with 10 bots per generation, and 100 moves per bot, per trial. The experiment is run in the dynamic and simulated condition.

Phase 2 of this project will transition into a Senior Thesis, and involve implementing WANNs using NEAT (Stanley and Miikkulainen, 2002).

*NEAT.*

*Analysis.*

**References**

Cappelle, C. K., Bernatskiy, A., Livingston, K., Livingston, N., & Bongard, J. (2016). Morphological modularity can enable the evolution of robot behavior to scale linearly with the number of environmental features. *Frontiers Robotics AI*, *3*(OCT), 1–10. https://doi.org/10.3389/frobt.2016.00059

Clune, J., Mouret, J. B., & Lipson, H. (2013). Summary of the evolutionary origins of modularity. *GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference Companion*, 23. https://doi.org/10.1145/2464576.2464596

Gaier, A., & Ha, D. (2019). *Weight Agnostic Neural Networks*. (NeurIPS), 1–19. Retrieved from http://arxiv.org/abs/1906.04358

Huizinga, J., Mouret, J. B., & Clune, J. (2014). Evolving neural networks that are both modular and regular: Hyperneat plus the connection cost technique. *GECCO 2014 - Proceedings of the 2014 Genetic and Evolutionary Computation Conference*, 697–704. https://doi.org/10.1145/2576768.2598232

Kashtan, N., & Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the United States of America*, *102*(39), 13773–13778. https://doi.org/10.1073/pnas.0503610102

Livingston, N., Bernatskiy, A., Livingston, K., Smith, M. L., Schwarz, J., Bongard, J. C., … Long, J. H. (2016). Modularity and sparsity: Evolution of neural net controllers in physically embodied robots. *Frontiers Robotics AI*, *3*(DEC), 1–16. https://doi.org/10.3389/frobt.2016.00075

Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, *10*(2), 99–127. https://doi.org/10.1162/106365602320169811, http://www.cs.ucf.edu/~kstanley/neat.html

<https://github.com/ahunteruk/RNeat/>

(Cappelle, Bernatskiy, Livingston, Livingston, & Bongard, 2016)

(Gaier & Ha, 2019)

(Kashtan & Alon, 2005)