Project
L.I.C.A.
Learning Integration Calculation Adaptations

Project Members

Director

Brenden Brusberg

Assistant Director

Emerson Chin

Assistant Developer

Joaquin Talon

Research Team

Jack Williamson

Raghuram Selvaraj

Erich Schwarzrock

Ben Dottinger

AJ Viola

Brendan Huff

Aesthetic Design Team

Kirk Brown Jr.

Brenton Bloomgren

More info on this project at https://github.com/BobJoeTom

HELLO!

Purpose

The goal of this project is to mimic the evolution of neural networks by replicating the results of the NEAT paper released by the University of Texas. We will recreate the algorithm and implement an evolution process of mathematical topologies that represent the logic trees of artificial intelligence. In theory this will show how a machine will learn by adapting and creating solutions to a problems.

Or simply create a game much like Asteroids were enemy spaceships are controlled by the computer and learn against your play style.

That is a bit of a mouthful... let's break it down

Brenden

Goal

Our goal is to properly use the scientific method by attempting to achieve past experimental results, then enhance their hypothesis and find even more data points. The main objective is to create a topological evolution algorithm compiled from past algorithms, naturalistic influences, and our own thoughts on how we can improve neural evolution.

1.

Emerson

Goal

Additionally, we aim to create a neuroevolution algorithm that can be widely implemented and utilized within this school's computer science department. Any problem that involves input and output and can be tested for a fitness value for optimization is within our scope for implementation of our derived algorithm.

2.

Ben

Goal

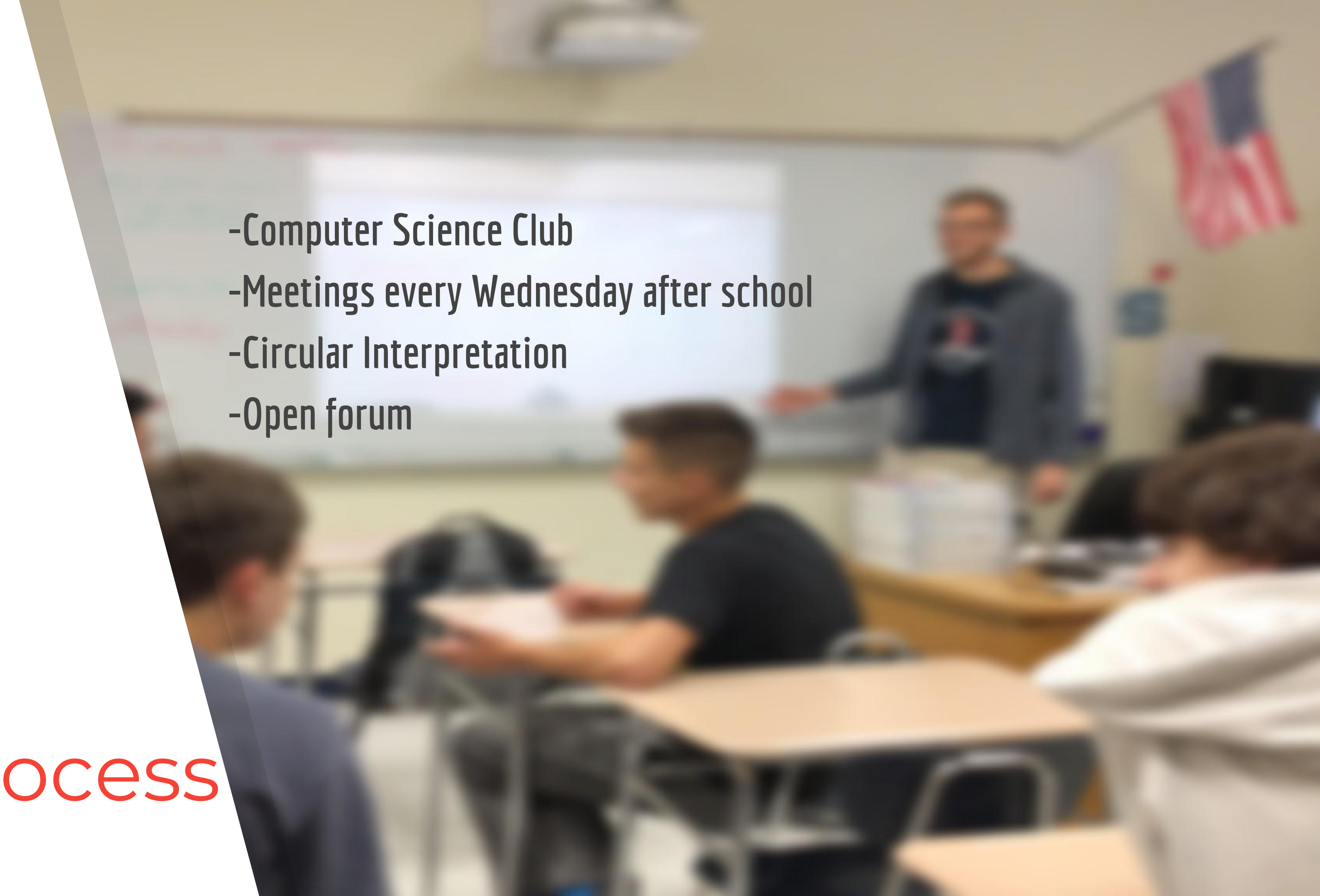
We plan to implement this algorithm in a game. This was the main purpose of our project: <u>to provide an interactive medium</u> that demonstrates the power of machine learning and AI to <u>better help people understand these concepts.</u> Machine learning and artificial intelligence are the future of how our economy and culture will operate. Through automation and computation, many unachievable goals will become achievable. We hope to contribute to the progress of machine learning through research and introducing people to AI.

We will also give the school a rendering engine. We are in the process of publishing our results online and making the game available on multiple websites

3.

Raghu & Brenden

The Research



-Computer Science Club

-Meetings every Wednesday after school

-Circular Interpretation

-Open forum

The Process

Here’s what we’ve found in our quest for knowledge…

https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html - General course and theory of neural networks; integration, depiction, algorithms

http://neuralnetworksanddeeplearning.com/ - “Free” online book of node construction, fitness values, bias functions (for/against), differentiating between shallow and deep neural networks & the challenges and merits of both

https://www.coursera.org/learn/neural-networks - Course about “neural networks and how they're being used for machine learning, as applied to speech and object recognition, image segmentation, modeling language and human motion, etc.” (As described by Geoffrey Hinton, a professor at the University of Toronto).

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html - Neural net theory, application, and limitations
https://techcrunch.com/2017/04/13/neural-networks-made-easy/ - Machine learning vs. network training

https://www.technologyreview.com/s/513696/deep-learning/ - General article on the wonders of deep-learining machines

http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf -

Why Neural Networking?

NNs versus other forms of artificial intelligence

Expert systems:

- Hard-coded, complicatedly branching decision trees
- Explicitly algorithmically defined
- Requires an expert in applied field, typically
- Requires manual consideration of all possibilities -- inflexible
- Fixed effectiveness/efficiency

For many applications, NNs are more effective than other forms of AI for less development effort.

Neural Networking:

- Only inputs, outputs, and particular coefficients are defined
- Implicitly algorithmically defined; requires no manual development
- Requires only knowledge of inputs and desired outputs
- NNs automatically interpret and consider any possibilities, acting appropriately
- Self-developing; efficiency and effectiveness increase

Other Genetic Algorithm Options Used to Evolve Neural Networks

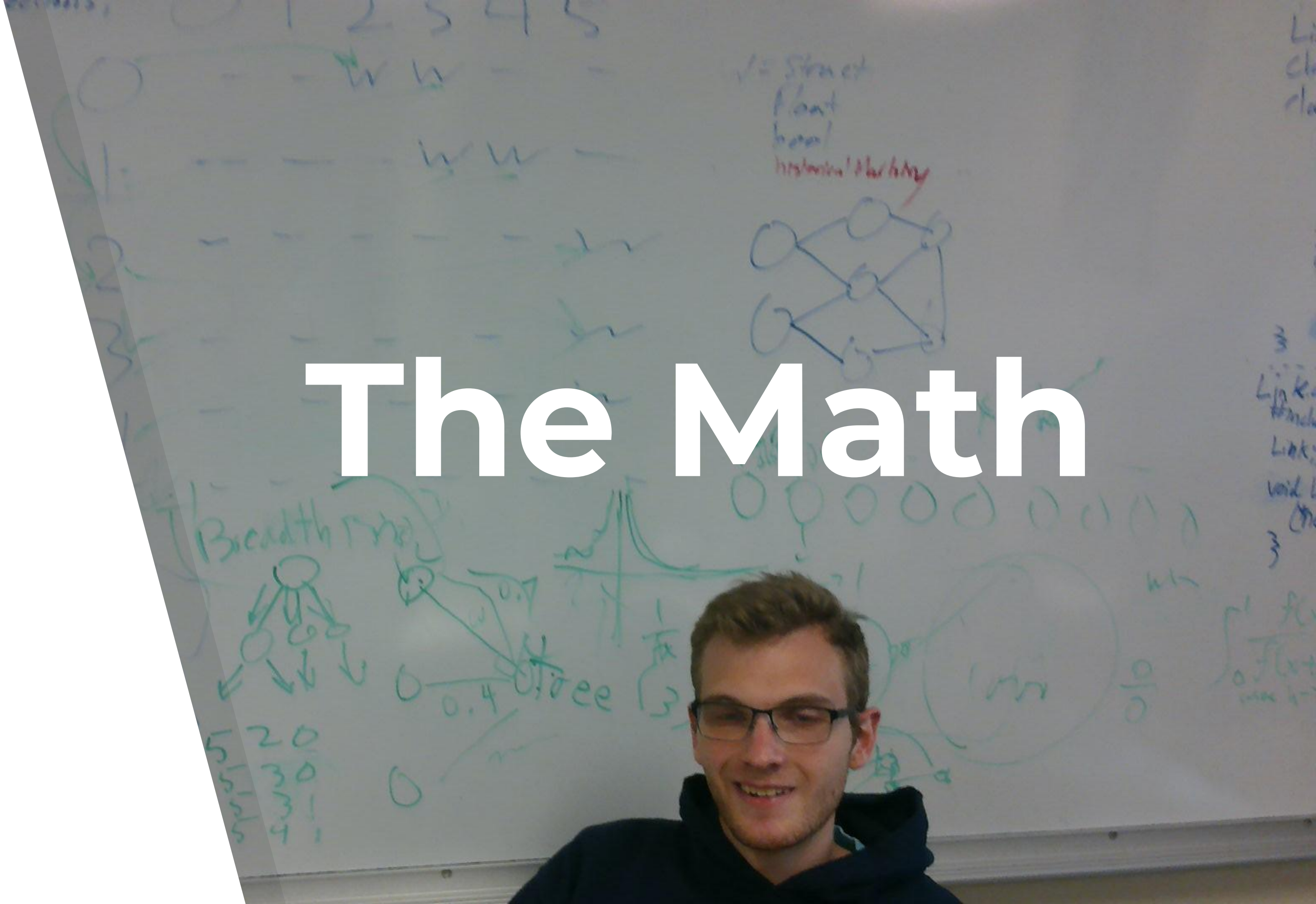
There are four other popular genetic algorithms that are can be used for evolving neural networks.

- Ev. Programming (Evolutionary Programming)
- Conventional NE (Neuroevolution)
- SANE (Symbiotic Adaptive Neuro-Evolution)
- ESP (Enforced Subpopulation)

All of these have been found to be less efficient or less effective than the NEAT(Neuroevolution through Augmenting Topologies) algorithm based on the number of generations or evaluations which is how to measure efficiency.

Method	**Evaluations**	**Generations**	**No. Nets**
Ev. Programming	**307,200**	150	2048
Conventional NE	**80,000**	800	100
SANE	**12,600**	63	200
ESP	**3,800**	19	200
NEAT	**3,600**	24	150

The Math



Genome Comparator

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \overline{W}$$

δ = compatability distance (optimizes gene mix)

E = # excess genes
D = # disjoint genes

represent genes outside of parent genome (matching or not)

N = # genes in the larger genome

$\overline{W}$ = average weight difference of genes

c_n = coefficient of factor importance

↳ helps to determine measure of fitness

Explicit Fitness Sharing

$$f_i' = \frac{f_i}{\sum_{j=1}^{n} sh(\delta(i,j))}$$

$$sh(\delta(i,j)) = \begin{cases} 1-\left(\frac{\delta(i,j)}{\sigma_s}\right)^{\alpha}, & \text{if } \delta < \sigma_s \\ 0, & \text{otherwise (else)} \end{cases}$$

σ_s = threshold of dissimilarity

σ_s represents number at which the species are genetically incompatible

α = constant

parameter which regulates the shape of the sharing function commonly set to 1 with resulting sharing ecr

- Ensures no one species gets too large
  - prevents one species from overtaking all others – genetic diversity.

i and j represent two organism types

$\delta(i,j)$

- represents the difference between organisms i and j

$\sum_{j=1}^{n}$

n represents total organisms j

j=1 assures that the species still exists

$\sum$ = total # sum of the organ

original fitness

$$\boxed{f_i = c_1 T (1 + c_2 H)}$$

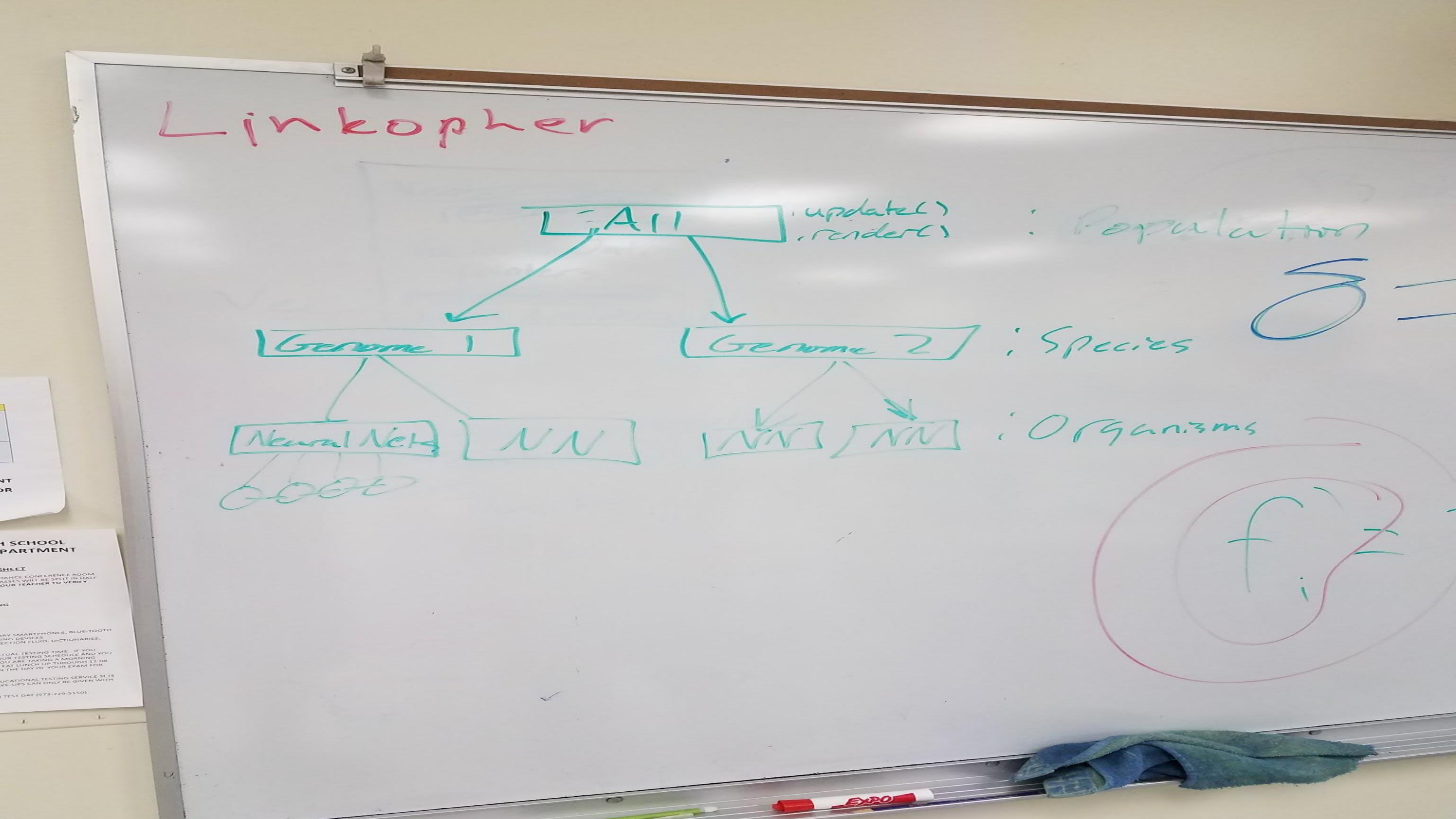
pull on weight

time alive

hits on player

f_i' = adjusted fitness for the new organism

f_i = original fitness value of the new org


Linkopher
All
:update()
:render()
: Population
Genome 1
Genome 2
: Species
Neural Nets
NN
NN
NN
: Organisms

es?

ght hp Times hit Player

hs Times Being Hit

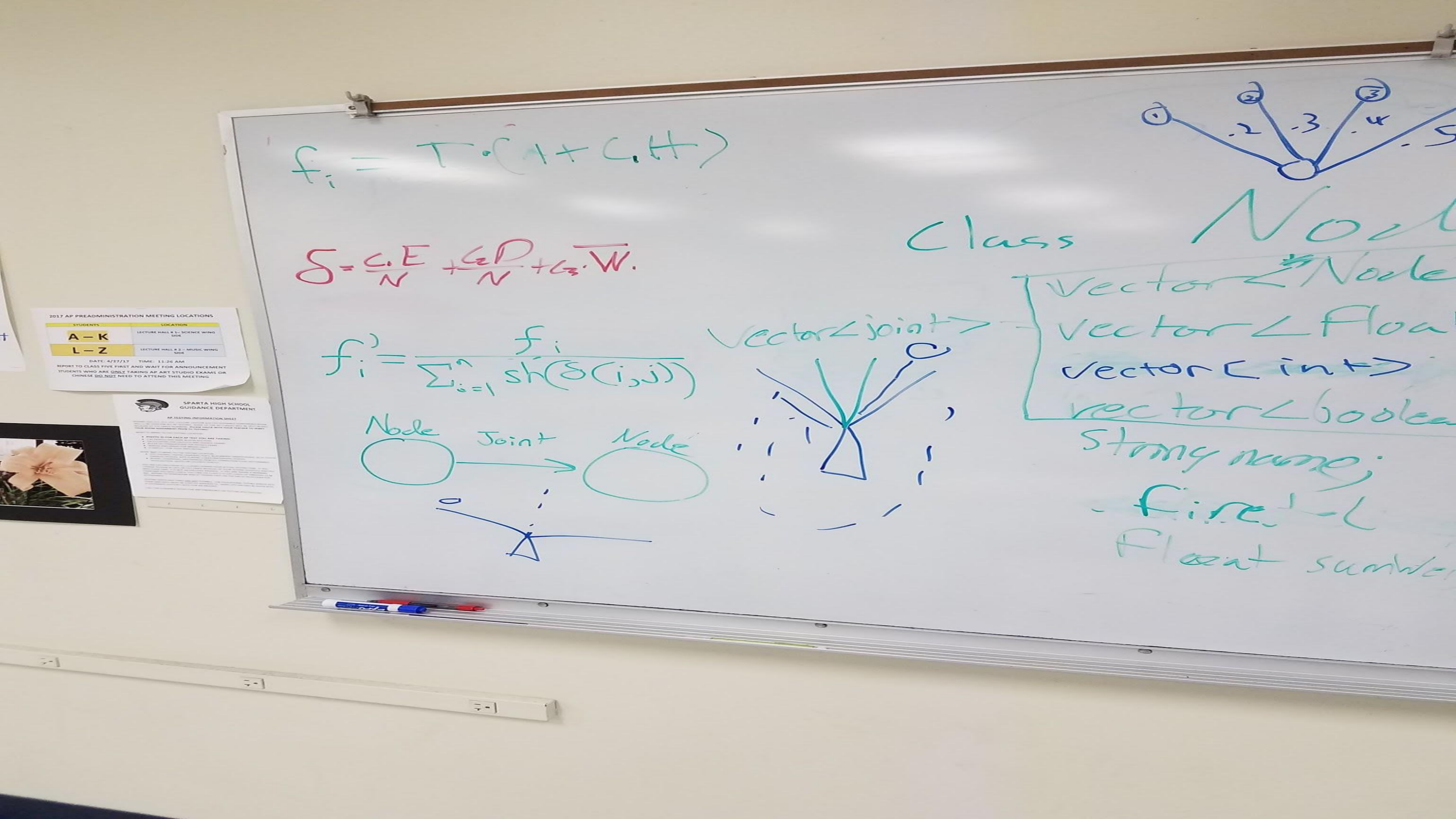
t Time Alive

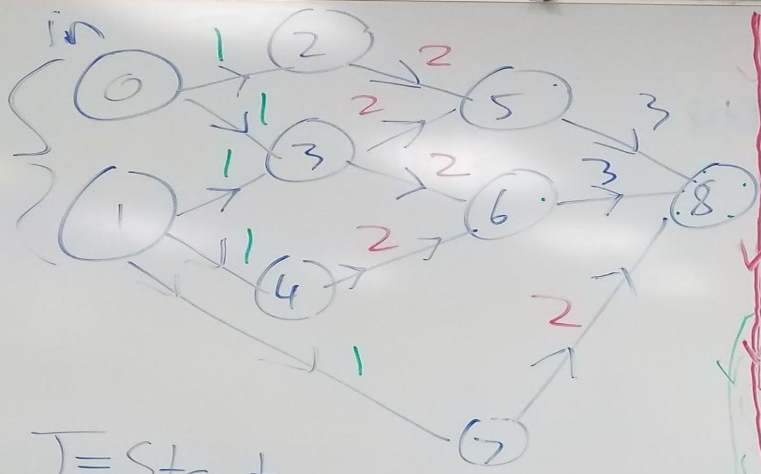
$$(h_p - h_s) + \left(\frac{t}{\text{weight}}\right)$$

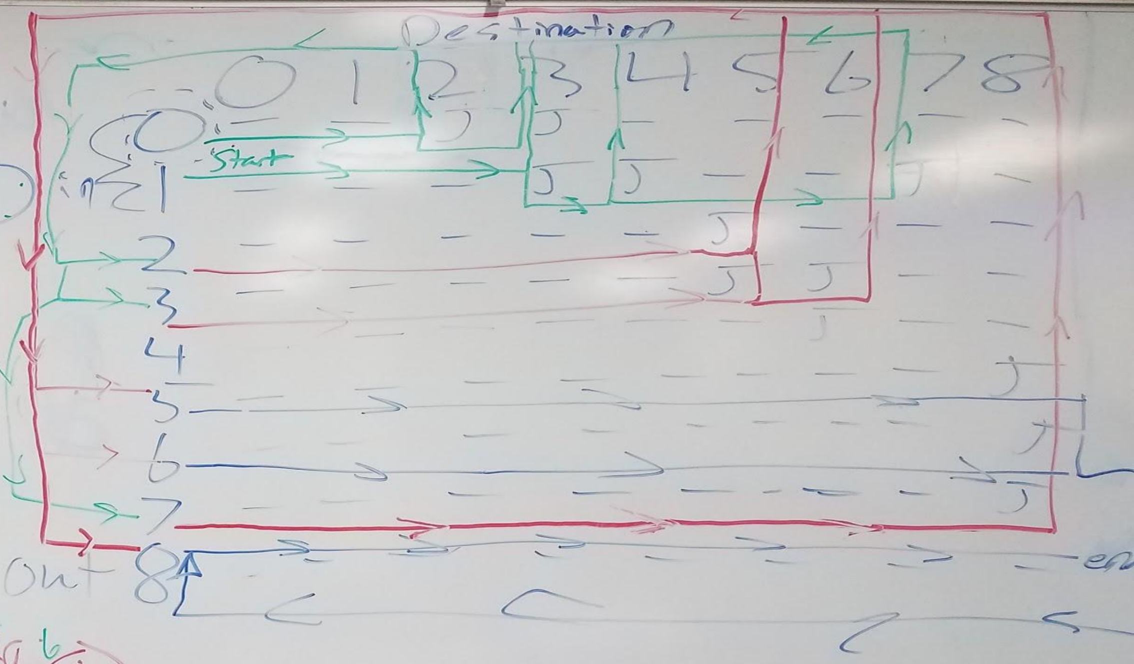
$$T(c_1 + c_2 h)$$

C_1 = so no multiply 0

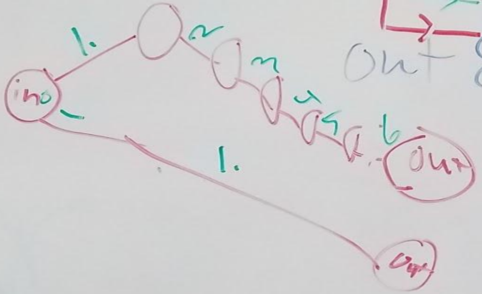
C_2 = hits bonus

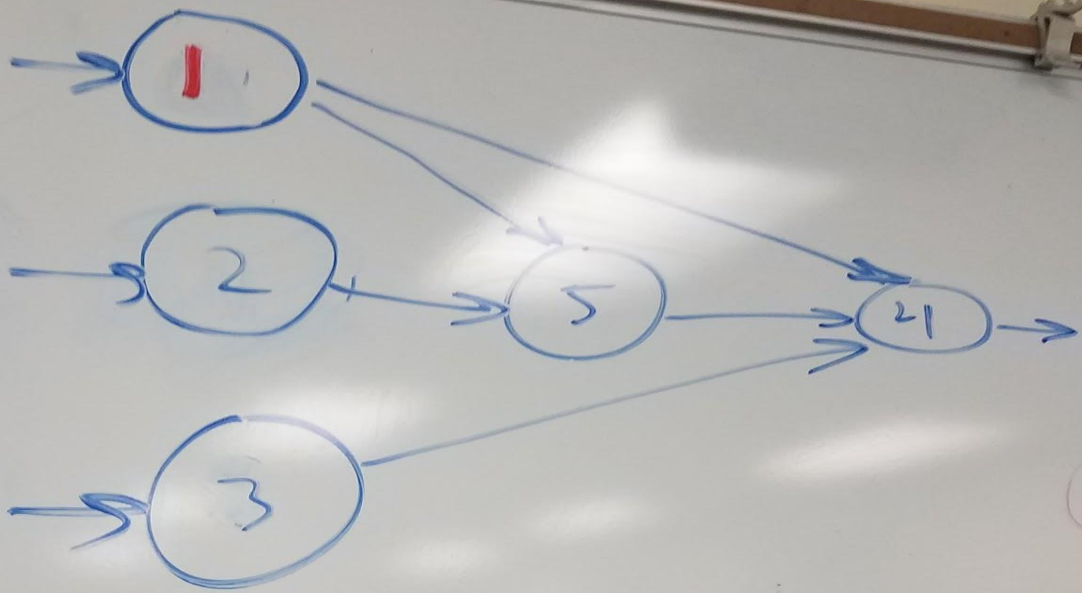

$f_i = T \cdot (1 + C_1 H)$
$\delta = \frac{C_1 E}{N} + \frac{C_2 P}{N} + C_3 \cdot \overline{W}$
$f_i' = \frac{f_i}{\sum_{i=1}^{n} sh(\delta(i,j))}$
Node
Joint
Node
vector<joint>
Class
vector<Node
vector<float
vector<int>
vector<boolea
String name;
fire
float sumWe
1
2
3
.2
.3
.4
2017 AP PREADMINISTRATION MEETING LOCATIONS
STUDENTS
LOCATION
A – K
L – Z
DATE: 4/27/17
TIME: 11:26 AM
REPORT TO CLASS FIVE FIRST AND WAIT FOR ANNOUNCEMENT
STUDENTS WHO ARE ONLY TAKING AP ART STUDIO EXAMS OR CHINESE DO NOT NEED TO ATTEND THIS MEETING
SPARTA HIGH SCHOOL
GUIDANCE DEPARTMENT





J=Struct
Float weight
bool enabled
int evolve#

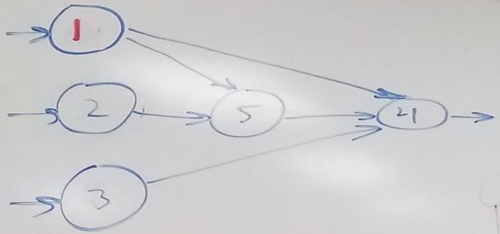


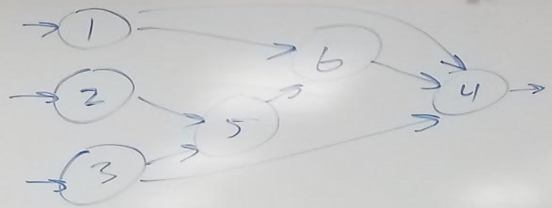


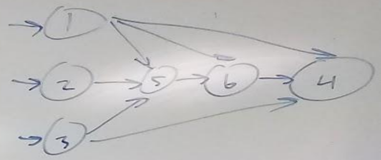
Disabled because of the addition of 5

Parent 1

1	2	3	4	5	8
1 → 4	2 → 4 Disab.	3 → 4	2 → 5	5 → 4	1 → 5







Disabled because of the addition of 5

Parent 1

1	2	3	4	5	8
1 → 4	2 → 4 Disab.	3 → 4	2 → 5	5 → 4	1 → 5

	1	2	3	4	5
1	—	—	—	J_1	J_8
2	—	—	—	J_2^D	J_4
3	—	—	—	J_3	—
4	—	—	—	—	—
5	—	—	—	J_5	—

Parent 2

1	2	3	4	5	6	7	9	10
1 → 4	2 → 4 Disab.	3 → 4	2 → 5	5 → 4 Disab.	5 → 6	6 → 4	3 → 5	1 – 6

	1	2	3	4	5	6
1	—	—	—	J_1	—	J_{10}
2	—	—	—	J_2^D	J_4	—
3	—	—	—	J_3	J_9	—
4	—	—	—	—	—	—
5	—	—	—	J_5^D	—	J_6
6	—	—	—	J_7	—	—

offspring

1	2	3	4	5	6	7	8	9	10
1 → 4	2 → 4 Disab.	3 → 4	2 → 5	5 → 4 Disab.	5 → 6	6 → 4	1 → 5	3 → 5	1 → 6

	1	2	3	4	5	6
1	—	—	—	J_1	J_8	J_{10}
2	—	—	—	J_2^D	J_4	—
3	—	—	—	J_3	J_9	—
4	—	—	—	—	—	—
5	—	—	—	J_5^D	—	J_6
6	—	—	—	J_7	—	—

Joint network[N][N] = {};
N = # of nodes

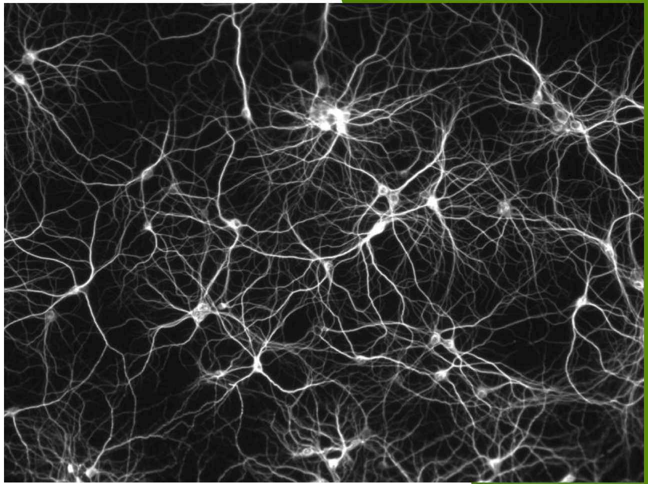
Calculating Compatibility

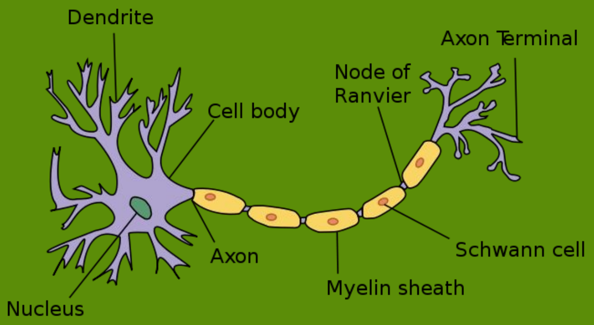
$$\delta = \frac{C_1 E}{N} + \frac{C_2 D}{N} + C_1 \overline{W}$$

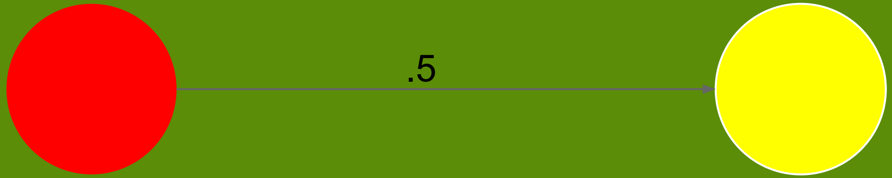
Calculating Fitness Value

$$f(i) = T_i(C_1 + C_2 H_i)$$

$$f'(i) = \frac{f(i)}{\sum_{j=1}^{n} sh(\delta(i,j))}$$

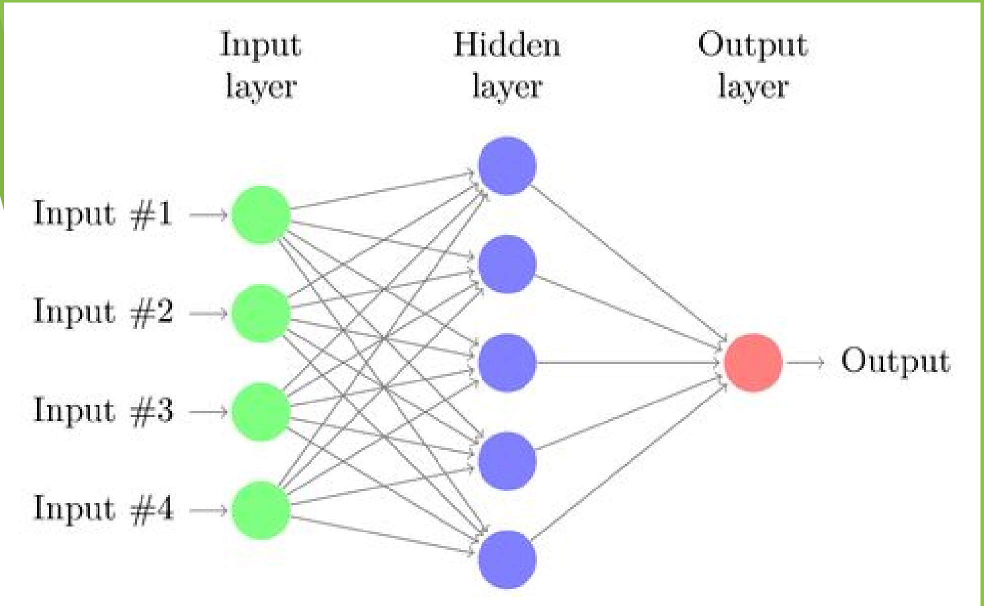






-Mathematical Topology
-Think of it as a map or structure
-Works by firing on an all or nothing premise
-Nodes consists of a sum weight value of all its inputs, once over 1.0 the node will fire all its outputs
-The joints between have a certain weight(0,1] which adds onto their connected node once fired
-They network becomes an advanced input output logic tree

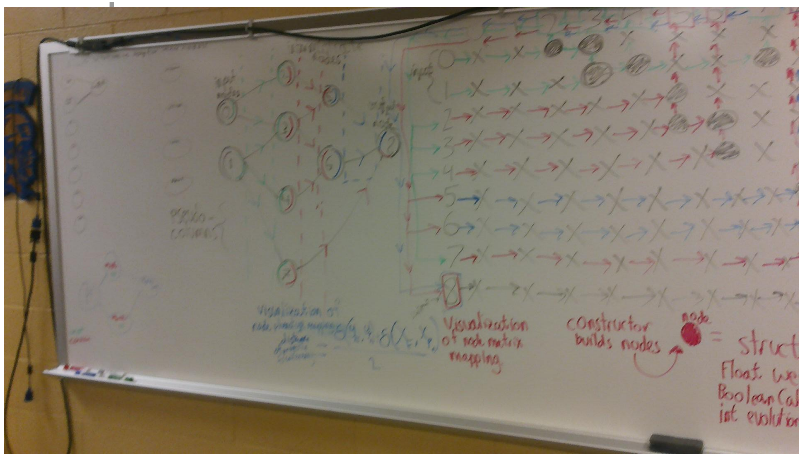
Input: Information about the player and obstacles

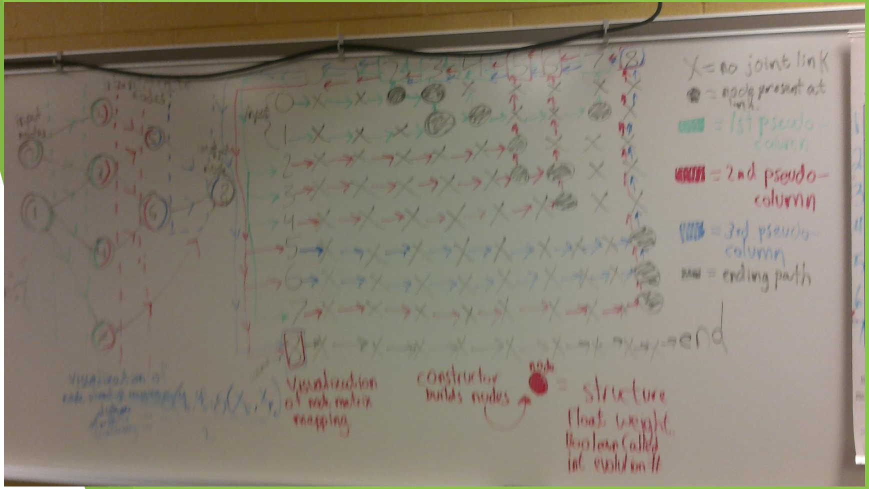


OutPut: Controlling the enemy players

Graph Theory

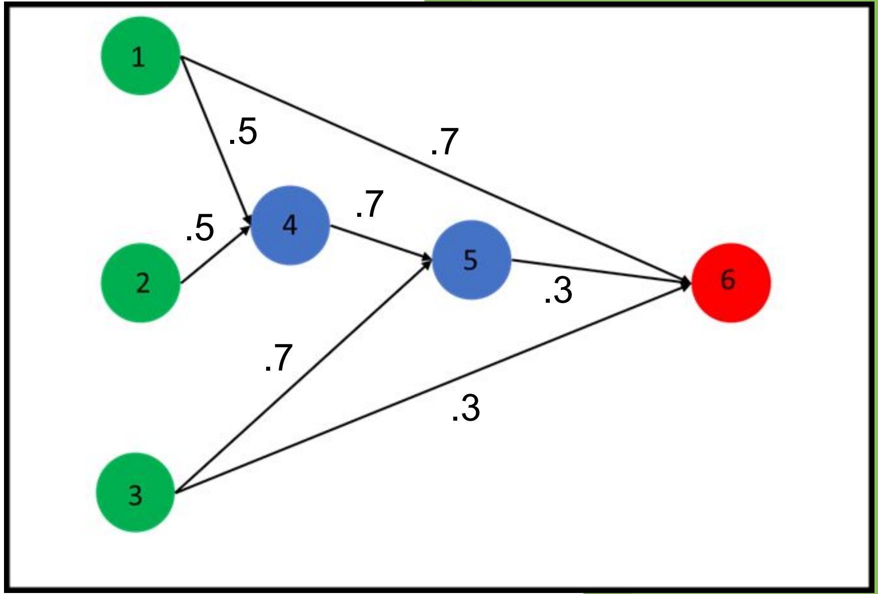
A visual representation of how the nodes are interconnected and how they activate other nodes in the





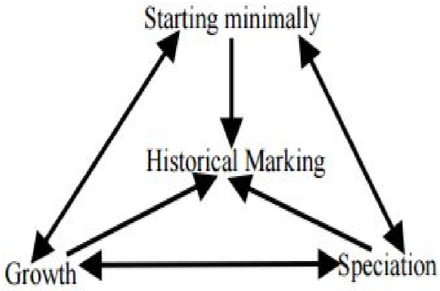
Implementing Graph Theory

How we draw out the connections between nodes within the system by way of two-dimensional vector systems



	Output						
	Node	1	2	3	4	5	6
	1				0.5		0.7
	2				0.5		
Input	3					0.3	0.7
	4					0.7	
	5						0.3
	6						

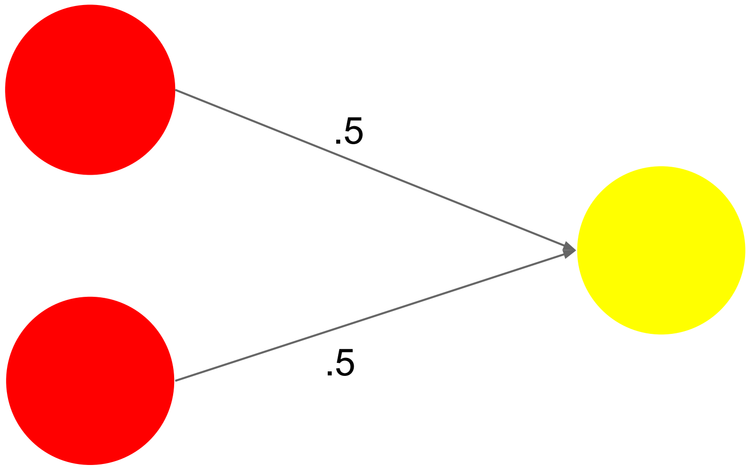
Evolution



Taking the top 50% of the simulated organisms

Types of Mutations

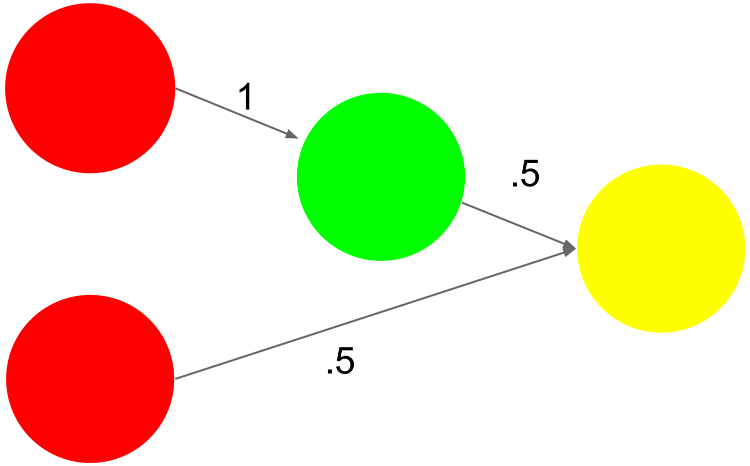
3 Types:



Types of Evolution

3 Types:

Adding a node

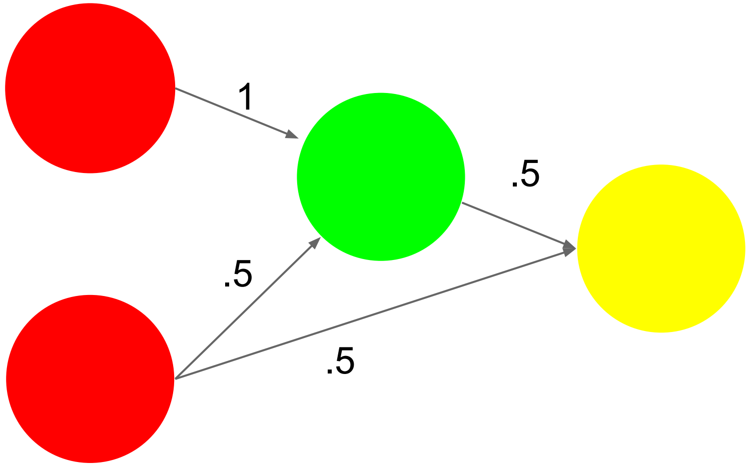


Types of Evolution

3 Types:

Adding a node

Adding a connection



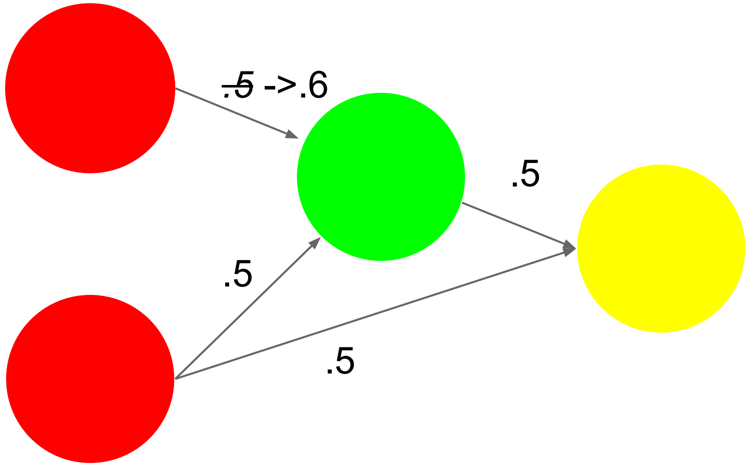
Types of Evolution

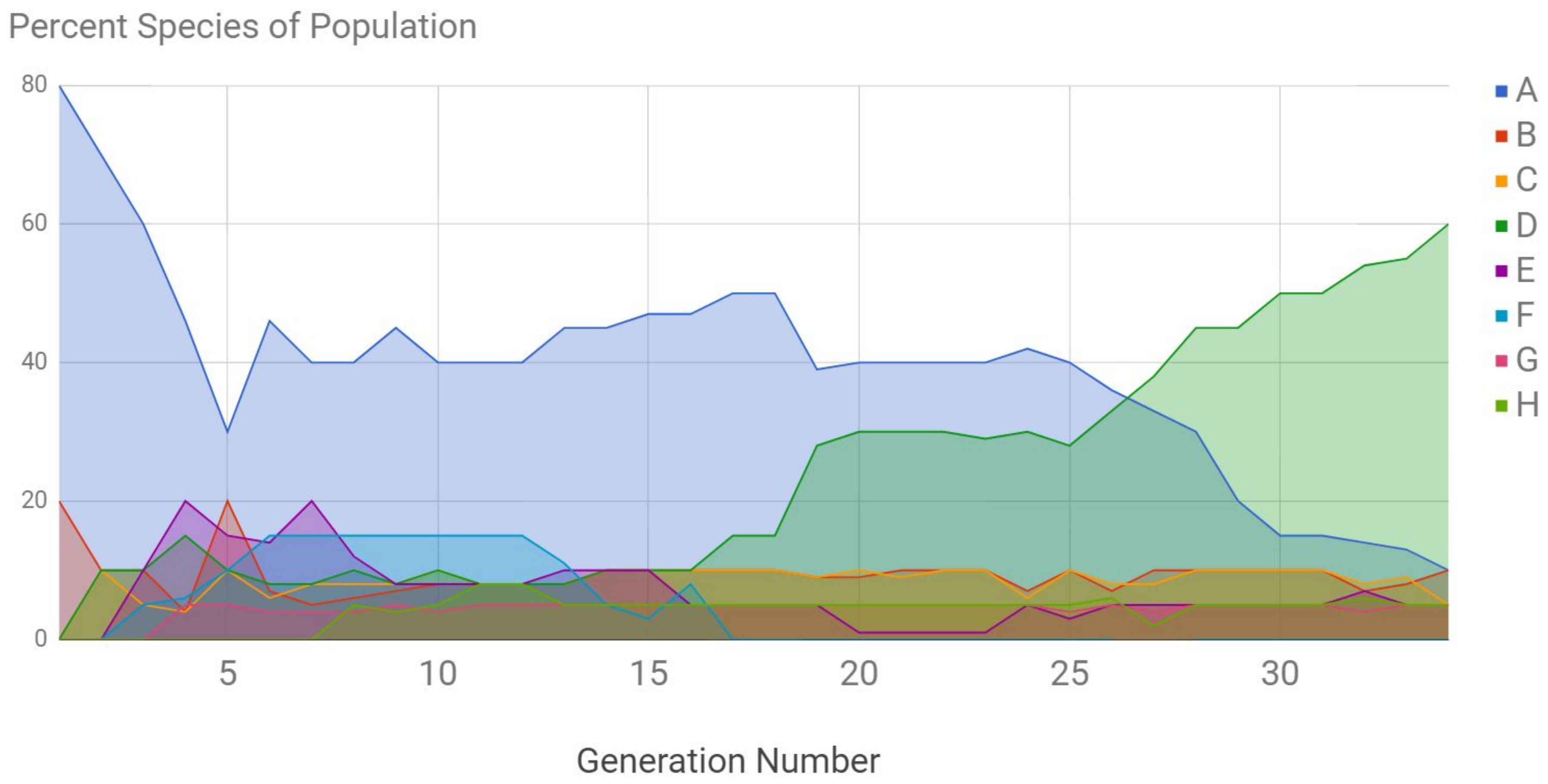
3 Types:

Adding a node

Adding a connection

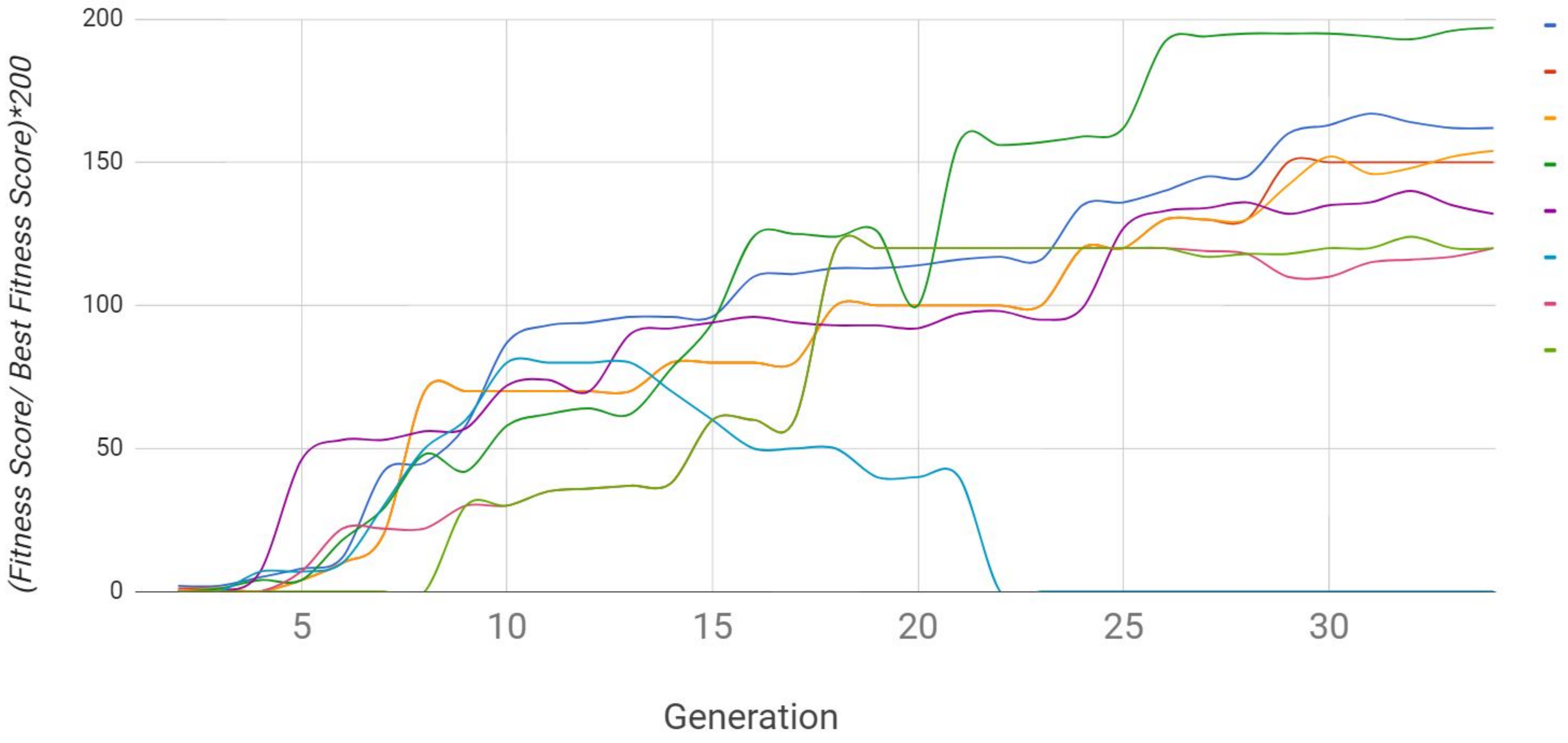
Adjusting a weight

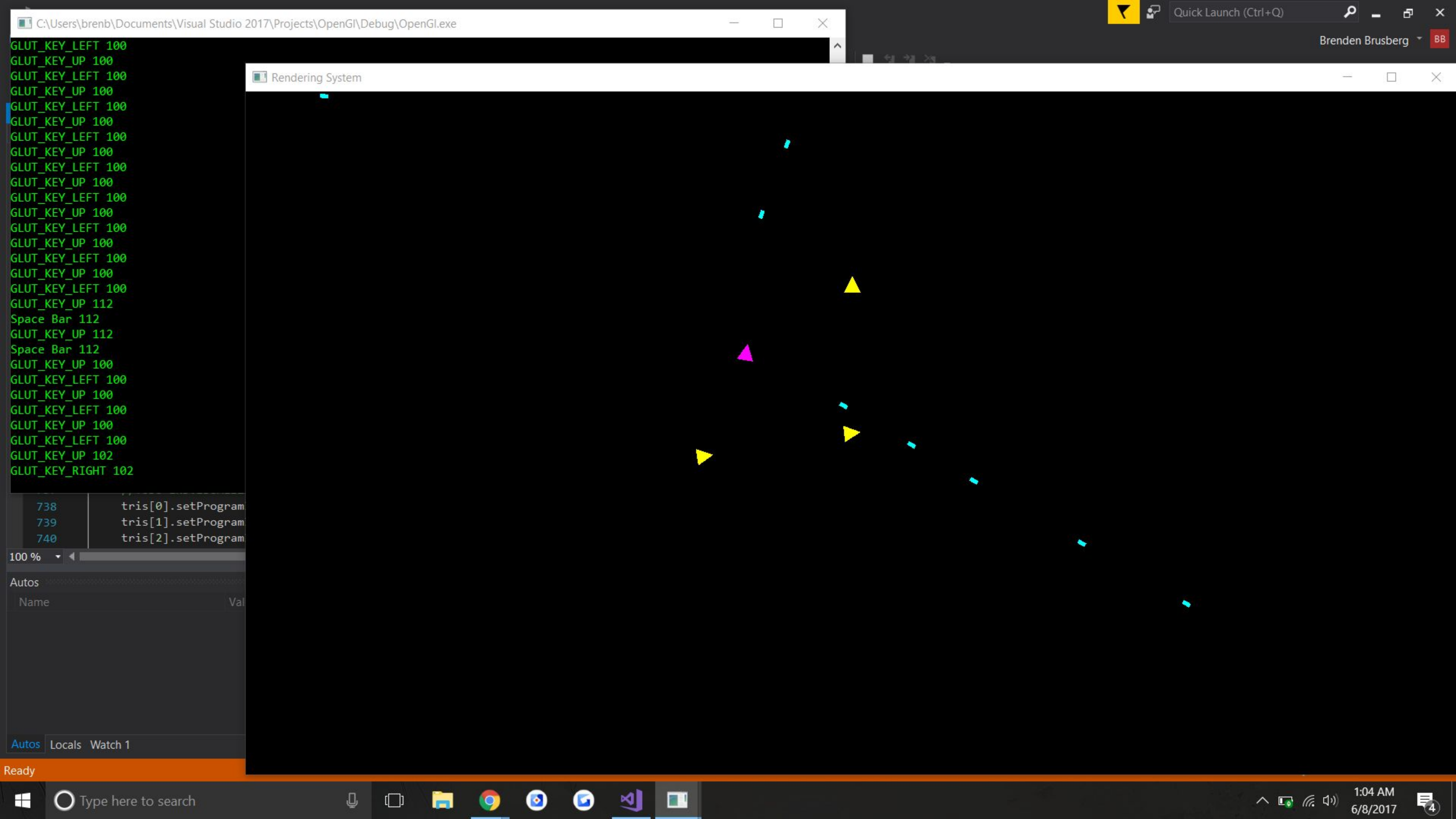




Percent Species of Population
80
60
40
20
0
5
10
15
20
25
30
Generation Number
A
B
C
D
E
F
G
H

Indvidual with Top Fitness from Each Species per Generation





C:\Users\brenb\Documents\Visual Studio 2017\Projects\OpenGl\Debug\OpenGl.exe
GLUT_KEY_LEFT 100
GLUT_KEY_UP 100
GLUT_KEY_LEFT 100
GLUT_KEY_UP 100
GLUT_KEY_LEFT 100
GLUT_KEY_UP 100
GLUT_KEY_LEFT 100
GLUT_KEY_UP 100
GLUT_KEY_LEFT 100
GLUT_KEY_UP 100
GLUT_KEY_LEFT 100
GLUT_KEY_UP 100
GLUT_KEY_LEFT 100
GLUT_KEY_UP 100
GLUT_KEY_LEFT 100
GLUT_KEY_UP 100
GLUT_KEY_LEFT 100
GLUT_KEY_UP 112
Space Bar 112
GLUT_KEY_UP 112
Space Bar 112
GLUT_KEY_UP 100
GLUT_KEY_LEFT 100
GLUT_KEY_UP 100
GLUT_KEY_LEFT 100
GLUT_KEY_UP 100
GLUT_KEY_LEFT 100
GLUT_KEY_UP 102
GLUT_KEY_RIGHT 102
Rendering System
738 tris[0].setProgram
739 tris[1].setProgram
740 tris[2].setProgram
100 %
Autos
Name
Val
Autos Locals Watch 1
Ready
Quick Launch (Ctrl+Q)
Brenden Brusberg
BB
Type here to search
1:04 AM
6/8/2017

“

How can a Neural Network be applied to everyday life?

Determine a problem where fitness is a factor

↓

Develop a Neural Network in a simulated environment to learn to deal with your problem

↓

Apply the network to your issue → solve your problem

Applying
Neural Networks

The applications of neural networks are near infinite -- anything with input, output and a goal in mind is feasible. To name just a few:

- Other strategy-based games (chess, etc.)
- Better & learning automated customer service
- Self-driving vehicles
- More efficient robot-assisted surgeries
- Simulation of evolution on earth
- Creation of artificial intelligence/robots that continually develop
- ***Creating works of literature by learning from society as a whole***

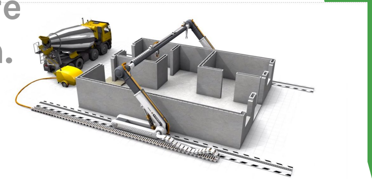
Some examples of problems that can be aided by a Neural Network:

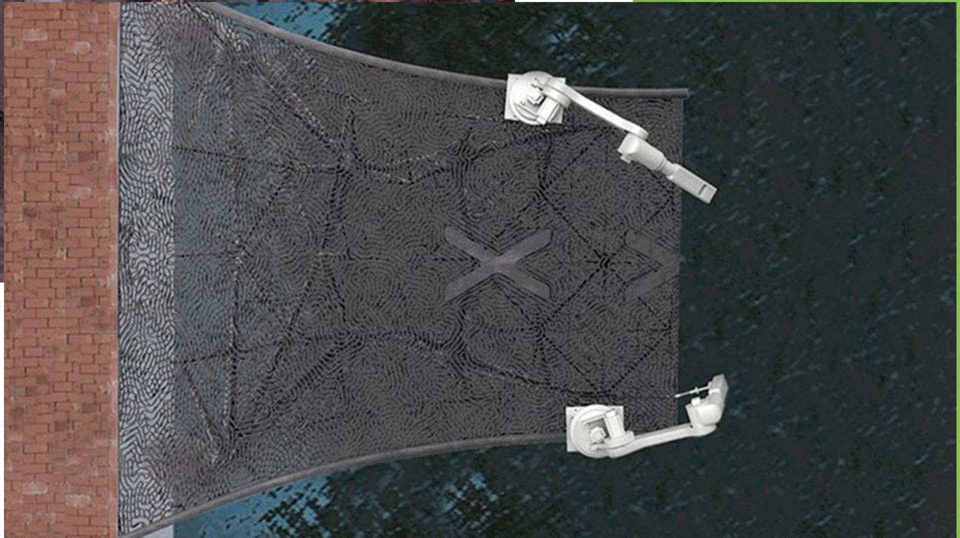
Medical Research:

- Mathematical topologies can take on the shapes of not only neural nets but also DNA or proteins.
- Also, in the search for cures for genetic disease, finding compatible plant DNA to replace sections of corrupted human DNA can be difficult. Using a neural network, scientists can weed out plants with traits that indicate incompatibility, reducing workload and speeding up the process of finding a cure.

Reliable transport, construction, security

- Imagine safer, faster, and economically-efficient transportation, construction, and security systems, all with computing infrastructure and effective integration.



CREDITS

Special thanks to all the people who made this possible

- Mrs. I for her time after school
- Doug Rudolph for sanity checks
- Ms. LeBlanc for a helpful analogy
- Mrs. Trzcinski, Mr. Hier, Mrs. Barker, and Mrs. Miller for classroom use
- Casa Capri for fueling our minds every Wednesday

THANKS!

Any questions?

You can contact the director at
brusbergb@students.sparta.org

or

brenbrus@gmail.com

Works Cited

Stanley, K., & Miikkulainen, R. (2017). Retrieved 5 June 2017, from http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf

Champandard, A. (2017). Reinforcement Learning Introduction. Reinforcementlearning.ai-depot.com. Retrieved 5 June 2017, from http://reinforcementlearning.ai-depot.com/

"Neural Networks". Doc.ic.ac.uk. N.p., 2017. Web. 6 June 2017.

https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

"Nielsen, Michael A. "Neural Networks And Deep Learning".

http://neuralnetworksanddeeplearning.com/

"Neural Networks For Machine Learning | Coursera". Coursera. N.p., 2017. Web. 6 June 2017.

https://www.coursera.org/learn/neural-networks

Works Cited (cont.)

"A Basic Introduction To Neural Networks". Pages.cs.wisc.edu. N.p., 2017. Web. 6 June 2017.

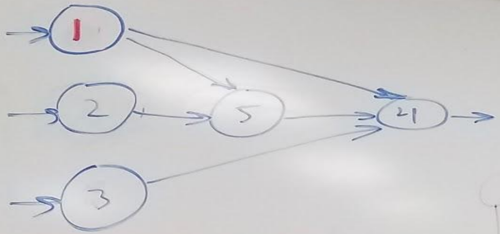
http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

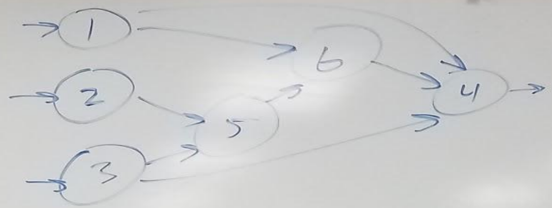
Tanz, Ophir, and Cambron Carter. "Neural Networks Made Easy". TechCrunch. N.p., 2017. Web. 6 June 2017.

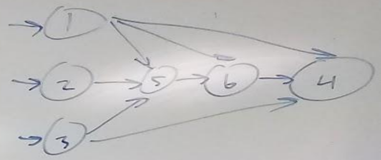
https://techcrunch.com/2017/04/13/neural-networks-made-easy/

Hof, Robert. "Is Artificial Intelligence Finally Coming Into Its Own?". MIT Technology Review. N.p., 2017. Web. 6 June 2017.

https://www.technologyreview.com/s/513696/deep-learning/







Disabled because of the addition of 5

Parent 1

1	2	3	4	5	8
1→4	2→4 Disab.	3→4	2→5	5→4	1→5

Parent 2

1	2	3	4	5	6	7	9	10
1→4	2→4 Disab.	3→4	2→5	5→4 Disab.	5→6	6→4	3→5	1→6

offspring

1	2	3	4	5	5	7	8	9	10
1→4	2→4 Disab.	3→4	2→5	5→4 Disab.	5→6	6→4	1→5	3→5	1→6

	1	2	3	4	5
1	–	–	–	J_1	J_8
2	–	–	–	J_2^D	J_4
3	–	–	–	J_3	–
4	–	–	–	–	–
5	–	–	–	J_5	–

	1	2	3	4	5	6
1	–	–	–	J_1	–	J_{10}
2	–	–	–	J_2^D	J_4	–
3	–	–	–	J_3	J_9	–
4	–	–	–	–	–	–
5	–	–	–	J_5^D	–	J_6
6	–	–	–	J_7	–	–

	1	2	3	4	5	6
1	–	–	–	J_1	J_8	J_{10}
2	–	–	–	J_2^D	J_4	–
3	–	–	–	J_3	J_9	–
4	–	–	–	–	–	–
5	–	–	–	J_5^D	–	J_6
6	–	–	–	J_7	–	–

Joint network[N][N] = {};
N = # of nodes