Welcome!

•••

To the BioMechatronics Club's
Genetic Algorithms Workshops

First Things First:

$ git clone http://github.com/dudds4/BioTronGeneticWorkshop -b tic-tac-toe

and open up the files Player.h and Player.cpp

Let us know if you have any trouble downloading the repo

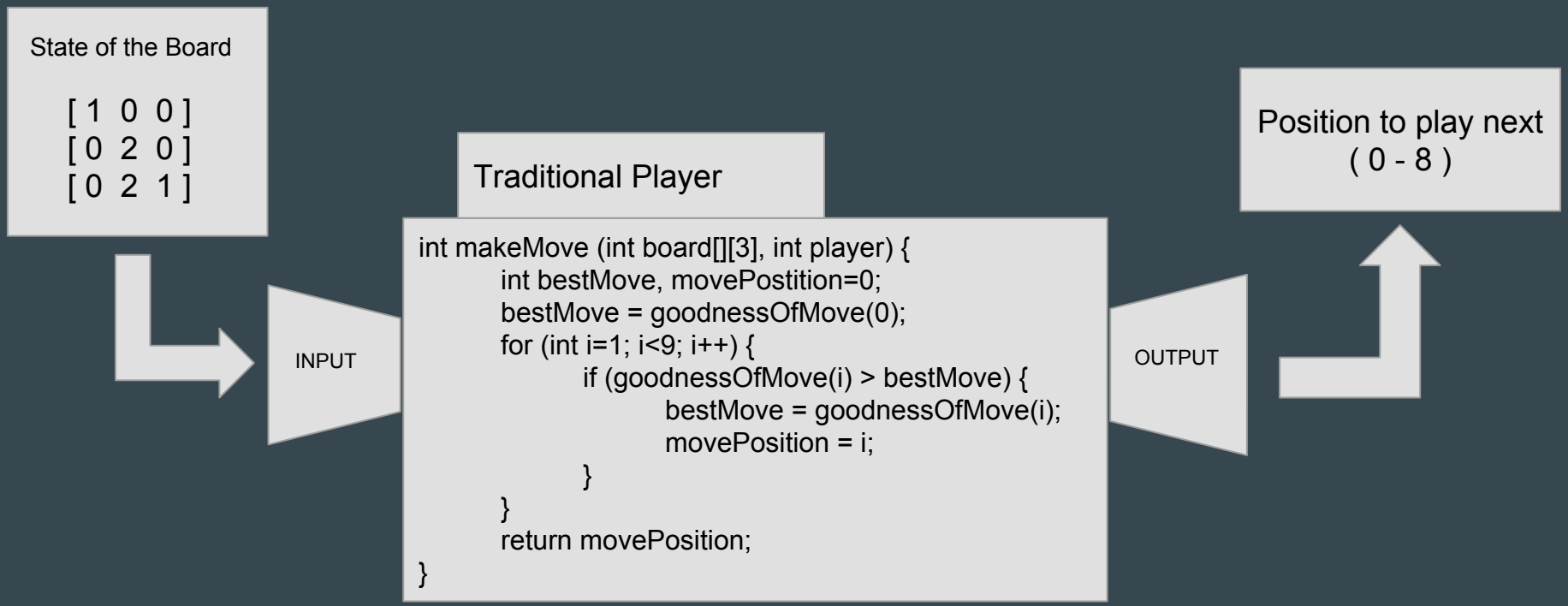
Artificial Intelligence

The “intelligence” exhibited by machines

Usually measured in the machine’s ability to replicate human intelligence

So what about tic-tac-toe?

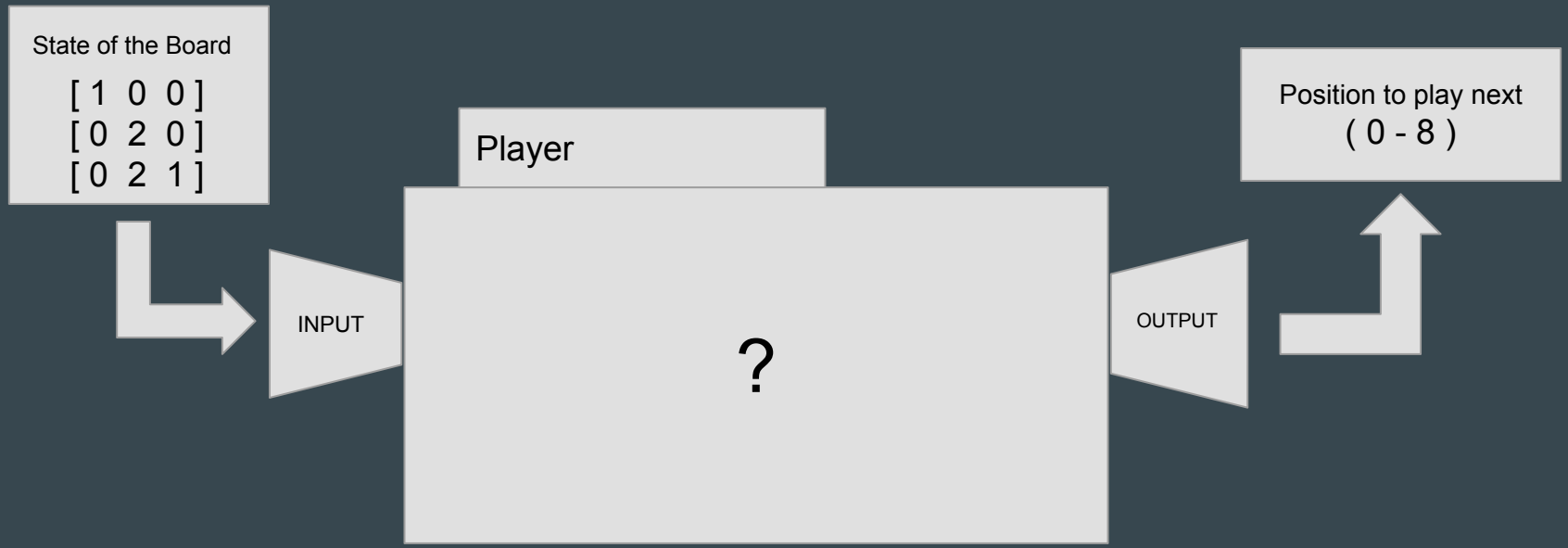
Traditionally we, as software developers, are tasked with creating a tic-tac-toe player and projecting our own intelligence onto it through our algorithm design



But this design is actually highly limited.

Machine Learning

So to move past these limitations, we're going to abandon the method altogether



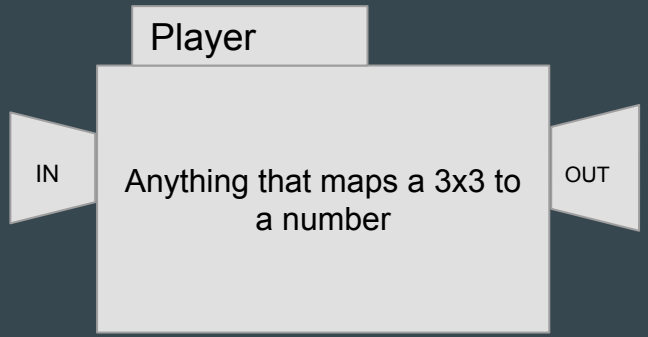
Machine Learning

This “operation” of converting a board state to a single integer can actually be represented in a number of ways:

-Matrix Multiplication

-Neural Networks

-Literally anything



Machine Learning: Pros and Cons

Traditional Method:

- Easy to read & understand
- Mimics human iterative thought process
- Difficult to create new instances of the algorithm
- Impossible to randomize and mutate
- Limited by human iterative thought process

Machine Learning: Pros and Cons

Data Structure Method:

- Effectively impossible to read or understand
- Very easy to create new instances of the “algorithm”
- Easy to randomize, edit, and iterate
- Limited by data structure complexity and computing power

Machine Learning

The act of editing and optimizing these data structures is what is meant by “machine learning”.

Depending on the process that is used to train the data structure and the complexity of the data structure itself, it could get arbitrarily good at whatever task it is being trained at.

Today we will be using genetic algorithms, the process of training your players through evolution.

Genetic Algorithms

The algorithm we will be using today will randomly generate a population of hundreds of unique tic-tac-toe players and it will have them play against one another.

Then, we will rank each player based on its skill.

The worst players will be deleted, and the best ones will be copied, mutated, and mated together to form the next generation of tic-tac-toe players.

Just as in biological evolution, natural selection will take place and after sufficient generations a “smart” player will emerge from the randomness.

Today's Workshop

What you will do today is define a Player, which can be literally any sufficiently complex data structure that takes in a board as input and maps it to a position to play at.

You will implement a makeMove function to take in the board and execute the mapping;

a mutate function which will make a small random change in the data structure;

and a mate function which will combine the properties of two Players.

Today's Workshop

Ways to do this:

- Traditionally
- Matrices
- Neural Networks
- A mix

Choose whatever you feel comfortable with and we'll all play each other at the end :)

Alternatives by level of experience

Less advanced:

> Check out the pre-implemented branch (rowan-tic-tac-toe or david-ttt) and modify constants to explore what factors (population size, number of generations, etc) generate better Players.

More advanced:

> Try Deep Neural Networks on the Connect4 branch (master)
>
> Modify Match and Trainer classes to train Player to do letter recognition on a 5x5