Dynamic Optimization Using Dense Neural Networks as a Compact Polygonal-Approximation of Constraints

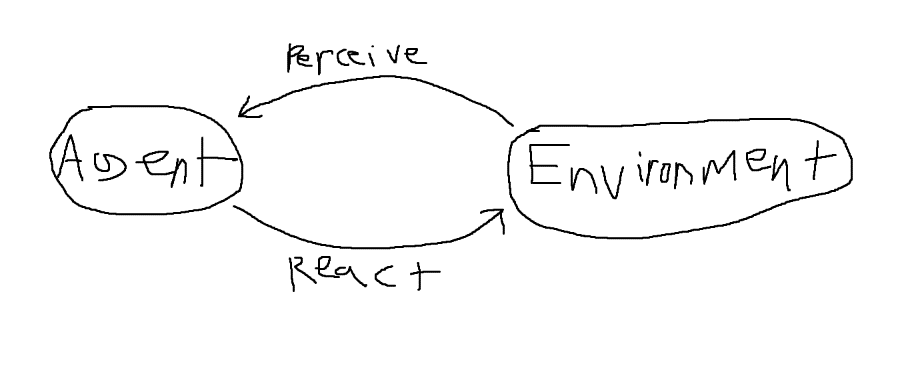
# Introduction

# Background

## Separators

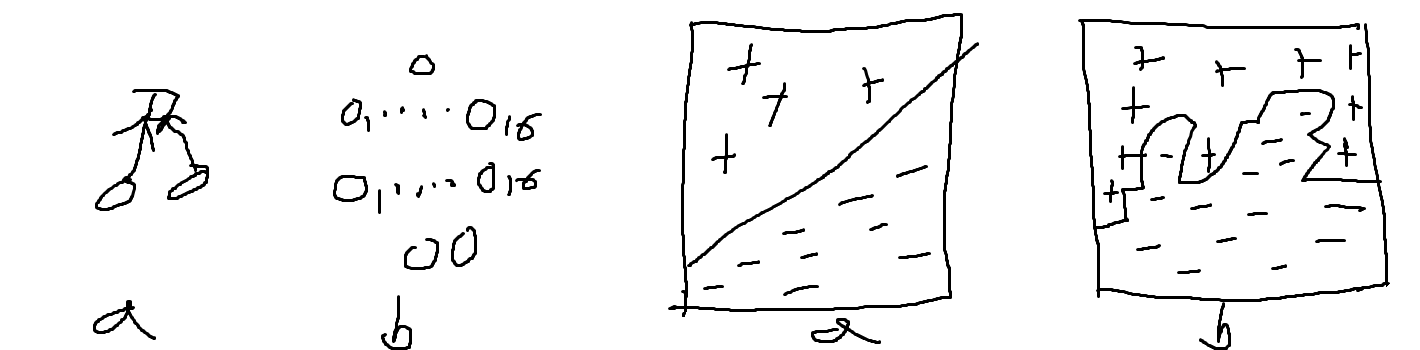
## Milling

# Proposed Methodology



We can view the agent and environment as separate entities. Furthermore, we shall take an ego-centric approach to the agent. Thus the environment does not affect the agent, but rather the agent perceives the environment and then acts upon the environment. The environment is seen as a passive and dynamic entity that the agent reads from.

The agent is composed of a learning module and an active inference module. The learning module takes pre-compiled samples that are generated from the environment and learns to differentiate between states that are considered “valid” or “invalid” according to criteria set by the engineering team. To accomplish this, a specific architecture is used to instantiate a neural network N. N is trained on the sampled data, such that it encodes the constraints defined in the environment which determine state validity.

This “valid” or “invalid” approach was chosen over other machine learning methods that are often applied to control problems, which generally involve predicting some function value like an expected future reward [citation here] or the optimal control signals of the system [citation here]. Each node in a neural network can be interpreted as a binary classifier, where a negative output identifies a point as negative and a positive output identifies a point as positive. A single node directly operating on two inputs is equivalent to a linear separator of the input space, as shown in Figure X. By successively stacking multiple layers of nodes, where each layer is followed by a ReLU activation function, then the decision surface is capable of a more complex polygonal separation as shown in Figure X.

Thus our neural network encodes both the separation of space as well as how to inference over this separation. Furthermore, once training is complete, each node will be normalized such that the magnitude of its weights is equal to one. The resulting network has output which can be interpreted as the linear pseudo-distance from the decision boundary. Pseudo-distance is used because the returned value is not the true Euclidean distance in the original problem space to the boundary, yet these pseudo-distances can be compared against each other in a way that is functionally equivalent to Euclidean distance [find source].

Thus any batch of localized points can have their pseudo-distances compared and processed

The agent wishes to avoid undesirable states while at the same time optimizing some set of properties.

## Simulator

The simulator serves as the host application for this paper in addition to

## References

[include SUMO citation for example of TCP connection through Unity]