VIGNETTE FOR PACKAGE YART

AT LEAST IT PRETENDS TO

Seminar: Solutions to All and Nothing
Sophism

Sebastian Sauer
My Immatriculation ID: 12345679
My Road 1, 12345 Somesmalltown

04. 10. 2019

Referee: Asc Prof. Oliver Obst

Abstract

Yart provides an RMarkdown template for rendering TeX based PDFs. It provides a format suitable for academic settings. The typical RMarkdown variables may be used. In addition, some variabels useful for academic reports have been added such as name of referee, due date, course title, field of study, addres of author, and logo, and a few more maybe. In addition, paper format (eg., paper size, margins) may be adjusted; the babel language set of Latex is supported. Those variables are defined in the yaml header of the yart document. Adjust those variables to your need. Note that citations, figure/ table referencing is possible due to the underlying pandoc magic. This template is not much more than setting some of the variables provided by rmarkdown (pandoc, knitr, latex, and more), credit is due to the original authors. Please reade the rmarkdown documentation for detailled information on how to use rmarkdown and how to change settings.

List of Tables

List of Figures

Contents

1 TODO

- Add a section that talks about the process of separating out a universal testing set and the principles/ideas behind this in data science.

2 Introduction

2.1 History of Robocup

2.2 Robocup Data

2.3 Quantitative Project Goals

Research questions here ## Background Theory ### Geometric Applications * Brief description of this method * Actually run the geometric application on the data * Show some diagrams * Give some results * State that these results will be used as a baseline to compare the neural network results with.

2.3.1 Neural Networks

- How Layers work
- Back Propagation
- Gradient Decent

2.3.2 Comparison of Geometry and Neural Networks

2.4 Previous Work/Literature

3 Methods

3.1 Applying a Neural Network to the same data as the Geometric Solution

The geometric solution to the problem of self-localisation has been obtained and is theoretically a perfect solution. This is not the case however, due to the perception error that is introduced by Robocup's simulation engine and as such the solution is as close as possible to the correct location. This means that this is the theoretical limit of accuracy for a neural network given the exact same input data. That is to say that given enough data, the function that the neural network begins to approximate, should be a good estimation of the geometric solution described above. If this is not the case and the neural network predicts the location of the agent better than the geometric solution, then there must be some form of systematic error being introduced by the perception error (as opposed to random error).

It is simple to apply the neural network to the same data as the geometric application, as inputs and outputs are of the same shape. There are some cases in which players can see less than two flags, and these cases are filtered as to not introduce any errors.

- Talk about the structure and training details of the network

3.2 Applying a Neural Network to data with more inputs

Mention using different neural networks for different input numbers.

The benefit of applying neural networks to this problem is that multiple inputs can be used to estimate the location of a player. This is helpful, as this can allow the perception error to be minimised, along with the possibility of unknown relationships in the data being identified and allowed for. It is for this reason that additional inputs will be added to predict for the same locations

An important note here is that in general, neural networks require a uniform input and output shape. This means that NaN values in the data can cause errors that break down the predictions provided by a neural network. This is an issue, as depending on the agent's position and orientation on the field, there are some instances where very few flags are observed. It is for this reason that multiple neural networks will be trained on the different cases of input numbers. That is to say that the data will be split up based on how many inputs are observed, and then individual neural networks will be trained for each of the splits. This will be capped above and only a **certain number (REPLACE THIS WITH SPECIFICS)** of observations will be considered.

3.2.1 Applying to More Inputs and Lagged Inputs

3.3 Hyper-parameter tuning

4 Results

4.1 Accuracy Tables

4.2 Visualisation of Errors

4.3 Results in Depth (If time allows)

This involves things like drilling into the accuracy of predictions by player number and the position that that player is on the field. It would allow for a nice visualisation that could overlay an accuracy heat map on a field.

5 Discussion

5.1 Comparison to Baseline

5.2 Comparison Across Projects

Whether the accuracy in prediction was higher in self-localisation than on player location. It should be expected that the self-localisation performs better, as there is more data to go on, without as much movement to worry about.

5.3 Combined Results

When the applications are stacked on top of each other, what will be the resultant compounded error? (this section should be relatively short as it would be expected that this is just an addition of errors multiplied by some scaling factor)

5.4 Applications/What does this mean?

Now that these predictions have been made, how can they contribute to the programming of the agents. How will this improve the play of a team

6 Future Work

6.1 Improvements

6.2 Extensions