Abstract

We present an investigation of two novel approaches to the problem of playing a virtual soccer game based on an unsupervised learning model of human player logs. EM categorization and a modified HMM algorithm are considered and an HMM is applied to a continuous state space / finite action space during a soccer game.

1 Introduction

One of the main proving grounds and incubators for Artificial Intelligence research these days is the application to games. There is an endless need for ever better virtual opponents to keep up with the entertainment demand for video games. In the past the definition of ‘better’ was that the computer opponent could beat the human player. More recently this has begun to be superseded by a concern with ‘realism’. An opponent which plays more like a human, with all the mistakes and sometimes erratic behaviour that implies, is more enjoyable to compete to with. The rise in popularity of massively multiplayer online games, where the majority of the game entities are human players connecting through the internet, may be partially attributable to this desire for better opponents. In this paper we describe one preliminary attempt to develop a usable method for creating realistic agents by using machine learning techniques to mine human player data. We investigate two methods for learning a model of how human beings play a virtual soccer game and make this the sole decision making engine of the automated player. The goal is to learn this model in an unsupervised way so that it is not necessary for someone to predetermine which information may be important or which types of strategies are preferable to others, the learning should be fully automated with no need to tune a large set of parameters or logical rules.

The methods looked at in this early stage of the research did not produce any positive results. They do however lead to several questions about the simplicity of the models used and to some ideas for improving on them in the future.
2 Background

2.1 Unsupervised learning

The original impetus for this project was to scale up the encouraging work done in [1] on unsupervised learning from player log data using a straightforward EM approach. They used a simple multi-room game that generated logs of human play. The main data object in their algorithm was a frequency matrix mapping each discrete state of the game to a discrete action. They then used a straightforward EM algorithm to cluster each game into a style of play that best explained its transition matrix. After these clusters were learned it was possible to generate play for any of the styles of play in a new game.

Our initial goal was to use the same strategy on a more complex game and see how well it does.

2.2 Virtual soccer

The game that we chose to work on is a soccer simulator called Tao of Soccer [2]. The program allows human players as well as programmed agents that need to conform to a simple API in java. There is a great interest in using soccer as a platform for AI research. The more widely known and used virtual soccer simulator for the RoboCup tournament is similar to Tao of Soccer except that it does not allow human play at this time and so would not usable for our approach.

Soccer is a complex enough game that linear, rules based solutions are non-trivial and usually not very realistic in comparison to how human’s play. Work on virtual soccer, largely using RoboCup, has often focused on reinforcement learning [3]. While the results from this are encouraging, they require a great deal of input from humans observing randomized agent gameplay and choosing good or bad strategies when they see them. As a general method for creating realistic agents this requires a lot of effort. Other studies [4] have looked at evolutionary techniques that reward ‘more fit’ players with the reproduction of their skills across the team. This also seems to work quite well but the definition of this fitness function is where the human bias comes in again to predetermine what realistic play will be. A more exciting solution would be to automatically generate a realistic playing style by mining human player data. If such an approach can be proven feasible for more complex games such as soccer then generating new, realistic opponents would be merely a matter of access to good data from human play.

3 Revised Approach

Our initial approach was to use the simple state-to-action mapping and EM clustering approach as in [1] but we soon realized that this would be unworkable. The original game used in the previous work had a very small number of possible states and so it was feasible to enumerate them all.

3.1 The curse of dimensionality

In the soccer game there are many variables that are continuous such as the position of the players or the ball. They are defined as decimal coordinates on a grid several thousand units across. Each state of the game is a permutation of these values. This approach quickly leads to having millions of states. This would produce an incredibly large and sparse transition matrix under the original algorithm which would make it unlikely any useful results would be produced.
3.2 Discretization

We considered the effect of reducing the state space by various methods for turning the continuous game data into discrete data. By dividing the field into just 3 zones and expressing angles to players and objects we would have been able to express the game with about 24,000 states. This is still large but may have been manageable, though it is questionable how useful such general states would be for making specific actions such as turning left or right or kicking at a particular moment.

3.3 Time

In addition to these problems there was the more fundamental concern that in soccer, unlike the multi-room game, there is a flow of actions that are important, that is, there is a notion of time and that certain actions are more or less likely to follow other actions. Even with the state space reductions the original model would not account for this feature of our game.

3.4 Hidden Markov Models

We decided at this point to consider a HMM style algorithm to model the games. The hidden states, linked one to another through a transition probability would be our actions (turn left, turn right, kick, go straight). The observed states, each linked to one hidden state by a gaussian probability, would be the state of our game at the current time step. Because HMM's can handle continuous or discrete observed states this allowed us to represent the states in their original form (although as we will see we later opted to reduce them again). The transition between actions would be calculated and summed over the play of the game and thus capture some notion of time in the game.

On top of this it turned out to be unnecessary to run the full EM algorithm multiple times in order to determine the appropriate clusters. The clusters under this scheme are the possible actions and each state observed while learning or predicting is determined with a certain probability to belong to each cluster. This is the probability of taking a particular action in that state of the game. While learning, we are in the lucky situation that the ‘hidden states’ are not actually hidden. The cluster that an observed state belongs to does not need to be estimated and maximized since we know it already, it is the action the human player took during their game. The distributions necessary to predict them, a probability transition matrix for actions and the parameters for the gaussians for the state to action distributions, can be computed in one step since the clusters are already fully known.

4 The Algorithm

Following the HMM approach for EM we compute the M step once to acquire our distributions and we perform the E step while playing to sample from those distributions and predict the best action.

4.1 M Step

We computed $a$, $\pi$, $\mu$ and $\Sigma$ after reading in the log file.

$$
\hat{\alpha}_{ij} = \frac{\sum_{t=1}^{T-1} \xi(x_t, i; x_{t+1} - j)}{\sum_{t=1}^{T-1} \gamma(x_t - i)}
$$

[5]
In our case $\xi$ is a way of choosing the action that was actually taken at that timestep rather than the probability of it. So for each timestep $t$ we increment the appropriate position in $a_{ij}$ for the actions that were taken at steps $t$ and $t+1$. After the game is read in we divide each row of $a$ by the total number of occurrences of action $i$.

$$\pi_i = \gamma(x_0 = i)$$

The initial state of soccer is not complicated, we set this to be the index for going straight.

$$\mu_i = \frac{\gamma(x_t = i) y_t}{\sum_{t=0}^{T} \gamma(x_t = i)}$$

These represent the centres of the gaussians that model each action in state space. To compute this we simply sum all of the states in the observed game that were matched with action $i$ and divide by the number of occurrences of each at the end.

$$\hat{\Sigma}_t = \frac{\sum_{t=1}^{T} \gamma(x_t = i)(y_t - \mu_i)(y_t - \mu_i)'}{\sum_{t=0}^{T} \gamma(x_t = i)}$$

Using a java matrix package we simply computed this covariance matrix at each timestep and added it to the appropriate entry in a sigma array for the action taken at that timestep then divided it at the end by the number of times that action occurred in the game.

4.2 E step

Armed with these distributions we now run the sampling algorithm while playing the game. At each timestep we compute $p(a_t | s_{1:t})$ the probability of each action given all states of the game so far.

$$p(a_t | s_{1:t}) \propto p(s_t | a_t) p(a_t | s_{1:t-1})$$

That is, the gaussian probability on the current state for each action and the probability of each action given all previous states of the game. To compute this we need:

$$p(a_t | s_{1:t+1}) = \Sigma_{1:t-1} p(a_t | a_{t-1}) p(a_{t-1} | s_{1:t+1})$$

Where this is evaluated with $a_t$ set to each of the possible actions. We compute this by keeping track of the sum from the previous timestep and adding to it this turn’s new lookup in the $a_t$ matrix and the previous turn’s answer from (1).

The best action with the highest probability after this calculation is the one that is sent to the soccer server for this timestep.

5 Results

We ran our tests on game logs generated with three players per side. We chose this to be as small as 3 in order to reduce the game to a more essential information set, reasonable behaviour with a small number of players would be achievement enough it seemed. We chose 3 rather than 2 due to the nature of the game being that if only two players are on the field the other default AI players (hardcoded with logical rules) would only designate a goalie if three players were on the team.
Given the complexity of the state space (22 dimensions consisting of x, y coordinates, ball possession and player facing orientation) and the fact that each action is modeled with one gaussian distribution we were beginning to expect it to be difficult for the system to pick out one action over another. Looking at the distributions that were generated it was clear that the µ points were all very different states but the variance values for most dimensions was extremely high. This is probably because states that ‘cluster’ to given actions are not necessarily close together with no states for other actions in between. Rather it seems that all states are spread amongst each other and the clusters all in fact over lap in many dimensions.

However in some dimensions this was not true, kicking for example was very strongly tied to ball possession so certain dimensions for the kick action had zero variance. This led to our first problem, that is, that the sigma matrix for the kick action was singular and could not be inverted as it needed to be for the gaussian calculation.

Ignoring kick for the time being we ran the system without it and found that the player had a strong penchant for going straight as is to be expected given that ticks pass 10 times a second and turns happens at most a few times every few seconds. The player initially turned left an inordinate amount of time and did not seem to react to the ball in any interesting way.

One other issue that seemed a problem was that the actual computed probability of each action at any moment was incredibly small on the order of $10^{-70}$. They were still ordered relatively so that the ‘highest’ one could be chosen but it would seem that this method has such a large spread across state space that any action at all is seen as incredibly unlikely. The implication of this is not yet clear.

6 Modifications

In order to attempt to improve upon the player’s less than stellar performance we tried a few modifications to our original strategy.

6.1 Angle to player/ball

Our first modification was to change the representation of the current state of a game from its original form of x and y position and orientation of all objects on the field to the angle from our player to each of these objects and the distance to them. This seemed somewhat more intuitive and hopefully less arbitrary to compare as positions.

The results unfortunately were not significantly different. The player still ran straight for a while, turned around then ran straight off the field until it hit the outer wall.

6.2 Relative angle to player/ball

Next we tried changing the angle to each object to be a relative one, relative to our player’s current orientation. So, for example, if the ball was in front of the player 15 degrees to its left the value for that dimension would be $-15$. We hoped this would vary more as the player moved and thus improve the chances of it reacting to its surroundings. It also could be argued that this is implicitly the information humans use when playing. It is less important to me what my absolute angle to the goal is than whether it is left of my current facing direction or right of it.
The behaviour of the player after this modification was quite similar to the previous modification.

### 6.3 Reduced state space

Fearing that somehow the sheer amount of extraneous data was the problem we opted to reduce the state space to 7 values from 22. Now a moment of the game would be described by the angle of our player to the ball and its distance from it, our player’s position and direction of movement and the angle and direction to the other team’s goal.

This had the positive effect that kicking was no longer a cluster with no variance, the distance to the ball when you kick it need not always be zero, as long as it is near enough to you to ‘possess’ it. Thus we could now invert the $\Sigma$ matrix for kick and include it in the calculations. We also found when running the player that it no longer had a larger tendency to turn left, it simply ran straight with the occasional left and right motions, which statistically is what should happen. However the player still showed no sign of actually reacting to events on the field, it simply ran straight across the field.

### 7 Future modifications

We believe the fundamental reason why these methods did not work is not because they are completely flawed but that they are too simple. Modeling the mapping of such a complex state space to actions with single gaussian distributions is obviously not expressive enough to capture the decisions being made. We would propose that future work attempt to do pre-clustering on each action to determine how many gaussian distributions would be needed to more reasonably model such a sparse and divided state space. Each action then, such as kicking, would not be a cluster but rather a set of clusters. A new state would be compared against all kick clusters and its probability of being there calculated. Any of these being the highest probability as compared to other action clusters would trigger the kick action. Only the API to the soccer program itself would need to know that this set of clusters all map to the same behaviour in the game. Hopefully this would produce some semblance of behaviour in the player.

Another possibility would be to forsake gaussians altogether and attempt to fit a multinomial or other distribution to each action. This of course would make later steps more difficult by requiring particle filters rather than an HMM.

### 8 Contributions of team

Myself and Tristram Southey worked together on this project from beginning to end. Most coding was done by the two of us sitting together, extreme programming style, writing the code, though I actually typed most of the time. Tristram is the soccer playing guru, having figured out how to play well, with the admittedly difficult Tao of Soccer interface, he generated the play logs. I wrote the parser that reads the log file and had to hack the game itself a bit to generate a usable action log and Tristram figured out innumerable ways as we went along of reducing our state space so that we continued to have hope that it might actually work.

### 9 Conclusion

Although it is very disappointing that the player does not react to events in the game I am satisfied at the end of it that we have taken this approach a fair distance and
removed our original doubts that it was not working because of our implementation. Seeing the data that was generated for the distributions seems to imply that this model really can only say that certain actions are likely to a certain degree on average rather than at a particular moment in the game. I think the granularity of one gaussian is the primary problem with the current approach. And I believe the current behaviour of the player is indicative of that. Switching methods halfway from the original EM to the HMM method and working on understanding the implications of both has been very rewarding even though neither worked. Exploring an algorithm in depth and searching for the problem with it has a way of exposing its true inner workings that isn't possible when it behaves close to what you expect right away. I look forward to further work on problems like this, where there is great possible benefit but an untried approach and hopefully time enough to explore the implications of the different tools at our disposal on the problem at hand.

References


