UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Heloisa Junqueira Barbosa

Aplicação de Machine Learning na identificação de jogadas no futebol de robôs

São Carlos

2022

Heloisa Junqueira Barbosa

Aplicação de Machine Learning na identificação de jogadas no futebol de robôs

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, para obtenção do título de Mestre em Ciências - Programa de Pós-Graduação em Engenharia Elétrica.

Área de concentração: Sistemas Dinâmicos

Orientador: Prof. Dr. Ivan Nunes da Silva

São Carlos

2022

Trata-se da versão corrigida da dissertação. A versão original se encontra disponível na EESC/USP que aloja o Programa de Pós-Graduação de Engenharia Elétrica.

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Barbosa, Heloisa Junqueira
B238a Aplicação de Machine Learning na identificação de jogadas no futebol de robôs / Heloisa Junqueira Barbosa; orientador Ivan Nunes Silva. São Carlos, 2022.

Dissertação (Mestrado) - Programa de Pós-Graduação em Engenharia Elétrica e Área de Concentração em Sistemas Dinâmicos -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2022.

1. Previsão de trajetória. 2. Previsão de caminhos. 3. Previsão de jogadas. 4. Sistemas Inteligentes. 5. Machine Learning. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE JULGAMENTO

Candidata: Engenheira **HELOÍSA JUNQUEIRA BARBOSA.**

Título da dissertação: "Aplicação de *Machine Learning* na identificação de jogadas no futebol de robôs".

Data da defesa: 24/08/2022.

Comissão Julgadora **Resultado**

Prof. Titular **Ivan Nunes da Silva** _____
(Orientador)
(Escola de Engenharia de São Carlos – EESC/USP)

Prof. Dr. **Danilo Hernane Spatti** _____
(Instituto de Ciências Matemáticas e de Computação/ICMC-USP)

Prof. Dr. **Claudionor Francisco do Nascimento** _____
(Universidade Federal de São Carlos/UFSCar)

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica:
Prof. Associado **João Bosco Augusto London Junior**

Presidente da Comissão de Pós-Graduação:
Prof. Titular **Murilo Araujo Romero**

AGRADECIMENTOS

Ao meu orientador, Prof. Dr. Ivan Nunes da Silva, por toda a confiança depositada ao longo destes anos.

À minha família e aos meus amigos, pelos conselhos e apoio que sempre deram força e possibilitaram a continuidade dos meus projetos. Em especial ao Rafael Lang, Arthur Demarchi e Guilherme Tavares pelo grande auxílio durante a proposta e desenvolvimento deste trabalho. Ao Leonardo Santos por seu essencial apoio e pela companhia e compreensão até nas horas mais difíceis.

Ao Warthog Robotics, por tudo o que me representa.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001

"Segue o teu destino,
Rega as tuas plantas,
Ama as tuas rosas.
O resto é a sombra
De árvores alheias. "
Fernando Pessoa

RESUMO

BARBOSA, H. J. **Aplicação de Machine Learning na identificação de jogadas no futebol de robôs**. 2022. 62p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2022.

Com o rápido desenvolvimento de novas tecnologias em software e hardware, tem-se disponível uma quantidade cada vez maior de dados provenientes de diversas fontes. Uma delas são as informações provenientes de dados sobre a localização de objetos, permitindo assim diversos estudos sobre trajetórias, tais como previsão, classificação, interação, dentre outros. Este trabalho baseia-se na teoria de previsão de trajetórias em multidões, regidos por um padrão e que decorrem da interação entre multi-agentes em um ambiente dinâmico. O ambiente escolhido para a aplicação desta proposta é o futebol de robôs, abordando a modelagem das jogadas consideradas com alto potencial de conversão em gols pelo time adversário. Nesta proposta foi realizadas a análises de 3 times do futebol de robôs: Ri-One, ITAdrois e Helios. Atingindo o principal objetivo deste trabalho, que é apresentar a proposta de uma ferramenta que por meio das classes de jogadas, a qual utiliza a modelagem através de máquinas de estados, e utilizando redes neurais será obtida a predição de jogadas no futebol de robôs.

Palavras-chave: Previsão de trajetórias, Previsão de caminhos, previsão de jogadas, Sistemas Inteligentes, Machine Learning.

ABSTRACT

BARBOSA, H. J. **Machine Learning applied in robot soccer game identification**. 2022. 62p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2022.

Due to the rapid development of new technologies in software and hardware, an increasing amount of data is available from different sources. One of which represents information from data on the location of objects, thus allowing several studies on trajectories, such as forecasting, classification, interaction, among others. This work is based on the theory of trajectory prediction in crowds, allied to the modeling approached in Multi-Agent Systems. The environment chosen for the application of this proposal is robot soccer, approaching the modeling of plays considered to have a high potential of scoring goals by the opposing team. In this proposal, the analysis of 3 robot soccer teams was carried out: Ri-One, ITAdrois and Helios. Achieving the main objective of this work, present the proposal of a tool that through the classes of moves, which uses modeling through state machines, and using neural network, is obtained the prediction of moves in robot soccer.

Keywords: Trajectory Prediction, Path prediction, Move Prediction, Intelligent Systems, Machine Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SMA	Sistema Multi-Agente
ML	*Machine Learning*, Aprendizado de Máquina
GPS	*Global Positioning System*, Sistema de Posicionamento Global
LSTM	*Long Short Term Memory*, Memória de Longo e Curto Prazo
RNN	*Recurrent Neural Network*, Redes Neurais Recorrentes
MDL	*Minimum Description Length*, Descrição do Comprimento Mínimo
DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*, Densidade Baseada na clusterização Espacial de Aplicações com Ruídos
ESN	*Echo State Network*
UDP	*User Datagram Protocol*
FSM	*Finite State Machine*, Máquina de Estados Finita
MAE	*Mean Average Error*, Erro Médio Quadrático
MSE	*Mean Squared Error*, erro médio absoluto

SUMÁRIO

1 INTRODUÇÃO

Devido ao rápido desenvolvimento de novas tecnologias em software e hardware, tem-se disponível uma quantidade cada vez maior de dados provenientes de diversas fontes, tais como a implementação de diferentes sensores em equipamentos, aparelhos que possuem *Global Positioning System (GPS)*, *smartphones*, dados provenientes do uso da internet, dentre outros (CAI et al., 2018). Esta grande quantidade de dados disponíveis possibilita a extração de informações que serão úteis para tomada de decisões em diversas áreas, tais como cálculo de rotas mais otimizadas, sugestões em anúncios on-line, identificação de pessoas, reconhecimento de voz, dentre diversas outras aplicações.

A disponibilização destes dados de uma maneira crua, ou seja, sem nenhum tratamento prévio, torna difícil a interpretação pelo ser humano, seja pela quantidade de dados, seja pela ordenação das informações disponíveis. Neste contexto lança-se mão de algoritmos que realizam o tratamento destes dados para então disponibilização dos mesmos para tomadas de decisão, análises, dentre outras aplicações, podendo realizar desde uma simples ordenação de dados, até tratamentos mais complexos tais como previsões futuras e classificação de padrões (YUAN et al., 2017).

Neste cenário, o acesso a diferentes dados sobre localização de objetos em movimento, tornou viável o desenvolvimento de pesquisas em diferentes áreas realizadas a partir de diferentes tipos de trajetórias. A principal fonte de dados quando se aborda o tema sobre trajetórias ou caminhos são trajetórias de pessoas e veículos, porém outros tipos de trajetórias são abordadas em diferentes pesquisas, tais como trajetória de rotas de aviões (YEPES; HWANG; ROTEA, 2007) (JACKSON; ZHAO; SLATTERY, 1999), marítimas(ULLMAN et al., 2006) (PERERA; OLIVEIRA; SOARES, 2012), de animais (LEE; HAN; WHANG, 2007), dentre outros. Tais dados podem ser utilizados para se obter uma grande quantidade informações tais como previsão de trajetórias de pessoas(KERFS, 2017), previsão de colisão em trajetórias(AMMOUN; NASHASHIBI, 2009), previsão de trajetória de furacões (LEE; HAN; WHANG, 2007), dentre outras. Tais análises poderão ser aplicadas em diversos cenários, tais como sistema de navegação de carros autônomos(JO et al., 2015), previsão do tráfego aéreo em aeroportos(MURCA et al., 2016), sistema de navegação de um veículo não tripulado (PREVOST; DESBIENS; GAGNON, 2007) dentre outras.

O cenário escolhido para aplicação desta proposta é o futebol de robôs simulado, o qual disponibiliza publicamente dados de jogos da categoria de simulação 2D da competição *RoboCup*. Organização esta de âmbito internacional criada com o propósito de proporcionar um ambiente para pesquisa focada em robótica e inteligência artificial, através um ambiente desafiador, dinâmico e de fácil implementação de soluções (REIS, 2003), sendo este a

principal característica deste estudo ser um ambiente dinâmico, onde há a interação entre os agentes e cujas ações são de difícil previsão.

1.1 Motivação e relevância do trabalho

Muitas são as abordagens sobre como extrair informações destas trajetórias, tais como clusterização, cuja finalidade é encontrar os principais caminhos através de padrões, e a predição, onde procura-se identificar as próximas coordenadas em determinada janela de tempo futura. Uma das abordagens relacionada a predição de trajetórias é levando em consideração a trajetória de outras pessoas como abordado em Xu, Piao e Gao (2018) e Alahi et al. (2016), sendo uma das abordagens que mais se assemelha ao objeto de estudo deste trabalho, ou seja a previsão de trajetórias em multidões.

Na abordagem utilizando a previsão de trajetórias em multidões é realizada a previsão do pedestre alvo em relação à interação dele com os demais pedestres. Em alguns casos como em Alahi et al. (2016) é utilizado o algoritmo *Long Short Term Memory (LSTM)* para cada pedestre, no qual os pedestres adjacentes compartilham camadas ocultas e consequentemente determinadas informações, realizando assim a previsão das trajetórias com base na interação entre os mesmos. A abordagem proposta se assemelha a estes trabalhos ao abordar a interação entre multidões. Porém neste trabalho os caminhos da bola somada ao dos jogadores irão constituir a classificação das jogadas de interesse como um todo, uma vez que realizará a classificação e posterior predição e que depende da interação entre os caminhos do time alvo do estudo.

Outra abordagem deste trabalho é quanto a interação abordada no conceito de Sistemas Multi-Agentes (SMA), consistindo na coordenação de agentes que apresentam comportamento autônomo, porém interagem entre si através da cooperação visando um objetivo final. Sendo um dos principais objetivos do jogo de futebol evitar levar gols do time adversário, esta proposta possui como principal foco, através da classificação e previsão de jogadas, identificar padrões considerados com alto potencial de conversão em gols por parte do time adversário e sendo assim prevê-los no decorrer do jogo. Uma vez que esta identificação possa servir de base em trabalhos futuros para a coordenação por modelagem, proposta em SMA, em ações que busquem evitar gols por parte do time adversário (REIS, 2003).

1.2 Objetivos e justificativa da tese

Na literatura são abordados trabalhos semelhantes, como em Asali et al. (2016) onde aborda a detecção da formação do oponente através da clusterização de caminhos e em seguida a utilização de rede neural para previsão das trajetórias do oponente, além do trabalho de Michael et al. (2018) o qual apresenta o aprendizado não supervisionado de uma rede neural através dos caminhos de jogadas selecionadas. A prosposta apresentada

através deste trabalho diferencia-se das demais, uma vez que propõe a identificação de classes de jogadas e através de uma rede neural supervisionada identifica a classe a qual pertence determinada jogada em andamento.

Para o desenvolvimento desta abordagem foram definidos 3 objetivos específicos:

Preparação do *dataset*: Nesta etapa são obtidos os dados provenientes do futebol de robôs simulado da categoria simulada 2D, em seguida são determinados os principais dados a serem utilizados e então gerado o *dataset* que contém o histórico de coordenadas das jogadas de interesse, sendo estas consideradas com alto potencial de conversão em gols;

Classificação das jogadas: Nesta etapa, de posse dos caminhos das jogadas de interesse, é realizada a clusterização dos mesmos, com a finalidade de se obter os principais caminhos. Uma vez que estes serão utilizados para a geração das cadeias da máquina de estado de todas as jogadas de interesse e posteriormente sendo agrupadas, de forma a se obter as principais jogadas. Então como saída final do algoritmo serão geradas as principais jogadas e suas classificações.

Predição das jogadas: Determinar em tempo real qual a jogada está em andamento, através da probabilidade da mesma pertencer a cada uma das classes obtidas na etapa anterior, e assim prever os estados futuros.

1.3 Organização do texto

No Capítulo 2 é apresentada o futebol de robôs e a teoria sobre SMA, no Capítulo 3 é apresentado a teoria e os trabalhos publicados a cerca de tais análises das trajetórias, uma vez que estes dois capítulos são a principal base teórica para este trabalho.

No Capítulos 4 e 5 são discutidas as principais ferramentas utilizadas para o desenvolvimento da proposta, no Capítulo 4 são apresentados os algoritmos de ML, introduzindo principalmente os algoritmos de Clusterização e Predição e no Capítulo 5 é abordada a Máquina de estados finita (FSM, em inglês), sendo utilizada principalmente na etapa de classificação das jogadas.

No Capítulo 6, a proposta deste trabalho é apresentada através do detalhamento dos algoritmos utilizados, no Capítulo 7 são apresentados os resultados obtidos e no Capítulo 8 são apresentadas as considerações finais e os trabalhos futuros desta pesquisa.

2 FUTEBOL DE ROBÔS E SISTEMAS MULTI-AGENTES

Os primeiros estudos utilizando o futebol começou a ser explorada por Mackworth (1993) trazendo uma análise sobre a robótica, onde cita o futebol de robôs como um dos cenários para aplicação da robótica. Já em 1993 Kitano, Asada e Kuniyoshi propuseram um programa de pesquisa utilizando o futebol para investigação no campo da robótica e IA, chamada *Robot J-League*, onde *J-League* era a liga de futebol profissional japonesa, atraindo o interesse também de pesquisadores americanos para tal iniciativa. Mas foi em 1995 que foi proposta a primeira *Robot World Cup Soccer Games - RoboCup*, competição esta de âmbito internacional (KUMMENEJE, 2003).

O principal objetivo da *RoboCup* é que através de um problema padrão, o qual também possui um grande atrativo ao público e a mídia, diferentes áreas e tecnologias sejam aplicadas para se atingir o objetivo de construir uma equipe, seja ela real ou simulada, jogando conforme as regras estabelecidas. Como também seja atingido o objetivo científico, que consiste em impulsionar a pesquisa no âmbito da robótica inteligente e da inteligência artificial (REIS, 2003)(ROBOCUP, 2018). Para tal o principal objetivo a longo prazo proposto pela *RoboCup* aos participantes, a fim de impulsionar tais objetivos, é: *"No ano de 2050, uma equipe de robôs autônomos humanoides, deve ser capaz de vencer a equipe campeã do mundo de futebol, num encontro disputado de acordo com as regras da FIFA."* (ROBOCUP, 2018).

A *RoboCup* além de ser composta pela liga *Soccer*, também é composta pelas ligas *Rescue, @Home, Industrial e Junior*, porém o foco deste trabalho é na liga *Soccer* onde possui as seguintes ligas: *Humanoid, Standard Plataform, Middle Size, Small Size, Simulation* (ROBOCUP, 2018).

Neste cenário do futebol de robôs será analisada a aplicação da teoria de SMA categoria Simulação sub-categoria 2D. O futebol de robôs simulado da categoria 2D consiste em 2 equipes, de 11 jogadores cada, que competem em um ambiente simulado, sendo este composto por 3 principais módulos, o servidor *(soccerserver)*; o visualizador *(soccer monitor)* e os logs dos jogos *(logplayer)*. Na Figura 1 é apresentado o ambiente de simulação 2D.

No ambiente de simulação o servidor é responsável por criar o ambiente virtual, ou seja o campo de futebol, e simular os movimentos de todos os objetos. Movimentos estes que são enviados pelos clientes e depois de executar tais movimentos envia a estes clientes a informação sensorial do campo, toda esta comunicação é realizada através de uma arquitetura cliente-servidor que utiliza *sockets User Datagram Protocol (UDP)*. O visualizador é responsável por se conectar, através de *sockets UDP* com o servidor, e

Figura 1 – Jogo da categoria Simulação da sub-categoria 2D.

Fonte: Elaborada pelo autor

permitir a exibição dos jogos através de um *display* gráfico, tornando possível a visualização dos jogos por parte do ser humano. Já o vídeo é uma ferramenta que permite a visualização de jogos que já ocorreram, pois todo o jogo que é simulado gera dois arquivos que contém informações relevantes sobre mesmo, um destes arquivos gerados é utilizado pelo *Logplayer* para que tais jogos, que já ocorreram, sejam visualizados e assim permitir a análise visual das estratégias dos times (ROBOCUP, 2019a).

Em SMA, agente é definido como um sistema de computador que possui duas principais características, a primeira é ser capaz de tomar decisões, até certo ponto, autônomas e a segunda é interagir com outros agentes através de ações análogas às atividades sociais, tais como cooperação, coordenação, negociação e outros (WOOLDRIDGE, 2009).

A partir do conceito de agente, definimos SMA como sendo dois ou mais agentes interagindo em conjunto e a partir da interação com o ambiente o agente tomará as ações necessárias de forma a atingir um objetivo ou determinada tarefa. Tais ações podem ser divididas em duas metodologias, sendo a primeira composta por agentes competitivos, consistindo na competição com agentes do time adversário, e a outra contendo agente cooperativos, consistindo na cooperação com agentes do próprio time (REIS, 2003).

Através das características descritas acima o ambiente do futebol de robôs apresenta diversos cenários para aplicação de problemas envolvendo SMA, sendo um ambiente dinâmico, não determinístico, com ruídos e agentes trabalhando para um objetivo final de vencer a partida, consistindo em um ambiente parcialmente cooperativo e parcialmente

adverso.

Muitos trabalhos foram publicados sobre ferramentas que utilizaram o futebol de robôs como plano de fundo e aplicando princípios de SMA, podemos citar Riedmiller et al. (2001) que aborda a investigação de algoritmos de *Reinforcement Learning* na coordenação de sistemas multiagentes no futebol de robôs simulado.

Como também abordado em Reis, Lau e Oliveira (2001) é proposto uma abordagem para estratégias em equipes cooperativas, englobando o posicionamento estratégico e troca de posicionamento, visando a aplicação da estratégia para evitar o conflito entre a reação do agente e a ação coordenada para a equipe. Podemos citar também o trabalho de Lang, Silva e Romero (2014), o qual apresenta uma proposta para o controle de sistemas de multi-agentes através de uma abordagem genérica, podendo ser aplicada em sistemas complexos, com agentes heterogêneos e controle distribuído.

Envolvendo a interação entre agentes é abordado por Reis (2003) a aplicação de diversos tipos de coordenação, tais como coordenação por percepção inteligente, por estratégia, por comunicação, por modelagem, dentre outras. Neste trabalho será abordada a teoria da coordenação por modelagem, que consiste na construção de modelos que auxiliem na tomada de decisão dos agentes no decorrer do jogo.

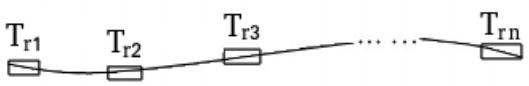
A abordagem realizada consiste na modelagem das jogadas consideradas com alto potencial de conversão em gols por parte do time adversário, servindo posteriormente para auxiliar na tomada de decisões na estratégia do jogo a fim de evitar legar gols por parte do time adversário.

3 PREVISÃO DE TRAJETÓRIA

3.1 Introdução

Neste trabalho será abordado a obtenção de informações provenientes da análise dos dados de coordenadas do futebol de robôs simulado. Para isto encontra-se na literatura a definição de trajetória, a qual consiste em um conjunto de coordenadas espaciais ordenadas pelo tempo, de um objeto em movimento como pode ser ilustrado na Figura 2, onde cada coordenada está representada por $T_{r_i} = (x_i, y_i, t_i)$, onde x_i e y_i representam respectivamente a coordenada x e y e t_i o tempo, e o conjunto destas coordenadas no tempo compõem uma trajetória, que é representada por $Trajetoria = (Tr_1, Tr_2, Tr_3, ..., Tr_n)$ (BIAN et al., 2018).

Figura 2 – Representação de uma Trajetória



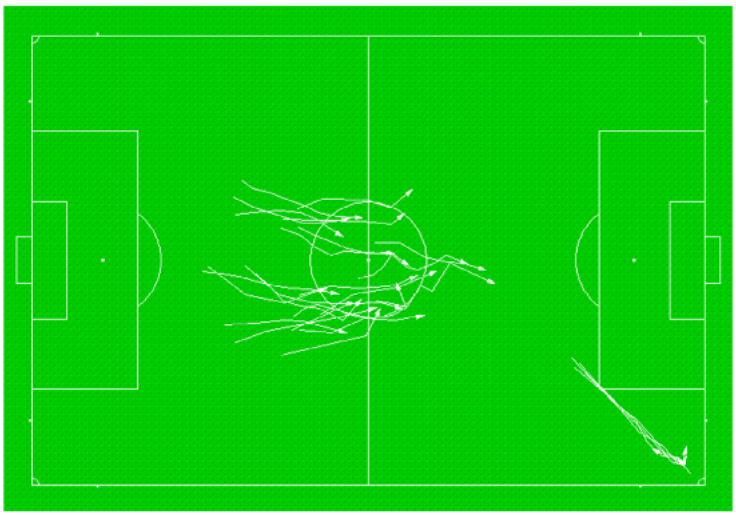
Fonte: (BIAN et al., 2018)

Neste trabalho ao utilizar estas coordenadas nos algoritmos de clusterização, a serem abordados nos próximos capítulos, serão utilizados somente os dados de coordenadas sem a sua dependência temporal. Sendo então aplicada a teoria de caminhos, onde o termo refere-se a sequências de posições que se seguem no espaço como apresentado em Flash (1987). Tal definição pode ser ilustrada através da Figura 3, onde são apresentados caminhos percorridos por jogadores, representados pela cor branca, como sendo uma sequência de coordenadas sem a informação temporal.

Esta definição de caminhos também é abordada em diversos outros trabalhos, tais como o tratamento das rotas de dados provenientes de GPS de motoristas, conforme apresentado em (QIAO et al., 2015), uma vez que a simplificação dos caminhos auxiliam na otimização do cálculo do tempo de execução das rotas de vários aplicativos baseados em localização, prevendo alternativas em situações críticas, tais como em congestionamentos.

Além das análises que fornecem informações para tomada de decisões, tem-se as abordagens que são diretamente aplicadas em robôs, desde os mais simples como em Yasutomi, Yamada e Tsukamoto (1988) que aborda a trajetória de um robô que limpa a casa. Até tarefas mais complexas, tais como em carros autônomos apresentado por Jo et al. (2015), onde para atingir seu principal objetivo de dirigir sem qualquer intervenção humana, precisa entender o meio ambiente em que está inserido, planejar sua trajetória,

Figura 3 – Caminhos percorridos por jogadores durante o jogo



Fonte: (GUDMUNDSSON; WOLLE, 2014)

evitar colisões com objetos e pedestres, dentre outros desafios, que um ambiente dinâmico possui.

Tais como os exemplos citados anteriormente, para a obtenção de análises mais complexas através de informações provenientes de trajetórias, lança-se mão de diversos algoritmos de ML e neste trabalho serão abordados dois principais tipos de algoritmos: clusterização e predição. Os algoritmos de clusterização são aplicados no pré-processamento de caminhos com a finalidade de gerar a classificação das jogadas de interesse, já os algoritmos de previsão serão utilizados na etapa final para a previsão das jogadas.

Na seção seguinte serão apresentadas aplicações de tais algoritmos na literatura, onde os mesmos apresentam um panorama de como tais algoritmos são abordados e quais as principais diferenças em relação a abordagem apresentada neste trabalho.

3.2 Trabalhos relacionados

3.2.1 Clusterização de Caminhos

A clusterização de caminhos é utilizada na etapa de modelagem do problema, uma vez que o tratamento dos dados provenientes dos caminhos dos jogadores de futebol de robôs, são constituídos de diferentes caminhos, de diferentes jogadores e da bola em diferentes jogos. Sendo composto por uma grande quantidade de dados faz-se necessário o tratamento prévio para obtenção dos dados. Como forma de sintetizar esta grande quantidade de caminhos serão utilizados algoritmos de clusterização, os quais possuem a característica de agrupar os diferentes caminhos, através de suas similaridades em diferentes núcleos (YUAN et al., 2017).

Os algoritmos de clusterização são aplicados em diversas áreas, utilizando diferentes

tipos de cálculo de distância para o agrupamento das mesmas. Tem-se por exemplo a clusterização de algoritmos realizada utilizando somente o cálculo das distâncias entre as caminhos, como em Buchin et al. (2018) onde foram utilizadas a clusterização dos caminhos através do cálculo da distância de Fréchet, a qual será apresentado com mais detalhes no Capítulo 4, posteriormente estes caminhos foram combinadas para a construção de mapas. Outros tipos de clusterização podem ser baseadas em diversos cálculos, tais como baseado em densidade para se obtenção padrões de operação em aeroportos como apresentado em Murca et al. (2016), classificação de trajetos utilizados por carros como abordado em Fu, Hu e Tan (2005), dentre outros.

Nos algoritmos anteriormente citados foram utilizados caminhos de forma contínua. Outro tipo de abordagem identificada na literatura é a clusterização feita a partir de sub-caminhos, ou seja é utilizado o particionamento do caminho em pequenos trechos e em seguida aplicado a clusterização para a obtenção dos principais caminhos, como abordado em Mao et al. (2017) e Lee, Han e Whang (2007), o qual consiste em particionar os caminhos, para então realizar a clusterização destes sub-caminhos utilizando o cálculo de agrupamento por densidade, tal abordagem visa obter a clusterização de partes do caminho, uma vez que quando utilizados os caminhos como um todo, não seriam identificados tantos detalhes em tais caminhos, tais como nos mais complexos.

3.2.2 Previsão de trajetórias

Após a etapa de agrupamento tem-se a etapa de previsão de trajetórias, uma vez de posse das trajetórias já identificadas e classificadas, as mesmas serão utilizadas no algoritmo de previsão, a fim de identificar jogadas específicas. Na literatura o conceito de previsão de trajetória é abordado em diferentes cenários, tais como a previsão de trajetórias envolvendo multidões (XU; PIAO; GAO, 2018)(ALAHI et al., 2016)(CHOI; SONG; YOO, 2019), trajetórias de voos (ZHANG; YANG; FANG, 2018)(YEPES; HWANG; ROTEA, 2007)(JACKSON; ZHAO; SLATTERY, 1999), trajetória e tráfego de veículos (LI; HE; ZHAO, 2010)(ZHANG et al., 2019)(WIEST et al., 2012)(AMMOUN; NASHASHIBI, 2009), trajetórias em alto mar (ULLMAN et al., 2006)(PERERA; OLIVEIRA; SOARES, 2012).

Um dos algoritmos mais utilizados na previsão de trajetórias são os algoritmos de Redes Neurais Recorrentes *(Recurrent Neural Network - RNN)*, que são redes neurais que possuem o aprendizado baseado em dados sequenciais e que armazenam informações ao longo dos ciclos (LIPTON; BERKOWITZ; ELKAN, 2015).

Um tipo de Redes Neurais Recorrentes é a arquitetura Memória de Longo e Curto Prazo (*Long Short Term Memory-LSTM*), como abordado em Choi, Song e Yoo (2019) e Xu, Piao e Gao (2018) onde este algoritmo é utilizado o na etapa inicial da previsão, com a finalidade de prever a posição futura de pedestres e em seguida são utilizados outros algoritmos, os quais consideram outras variáveis além da trajetória em si, para então

realizar a previsão final da trajetória. Em Alahi et al. (2016) é proposto uma variação da a utilização do algoritmo LSTM, na qual possui a característica de compartilhar camadas escondidas com os seus vizinhos, uma vez que tais camadas escondidas compartilham outras informações entre os algoritmos de LSTM, a fim de se obter uma previsão baseada na interação em multidões.

Dentre outros algoritmos utilizados para tal finalidade, podemos citar Modelos de Markov utilizado em Qiao et al. (2015), que durante o pré-processamento dos dados é aplicado algoritmo de clusterização baseado em densidade e em seguida é aplicada a abordagem chamada de predição de trajetória baseada em modelos ocultos de Markov, uma vez que este algoritmo é aplicado com a finalidade de obter informações importantes através das camadas ocultas, informações estas que não podem ser facilmente obtidas através das abordagens clássicas de predição.

A aplicação de tais algoritmos na previsão de trajetórias em multidões se assemelha a abordagem apresentada neste trabalho, uma vez que a previsão das trajetórias é realizada levando em consideração informações compartilhadas com os demais vizinhos, conceito este que será abordado com mais detalhes no Capítulo 4.

3.3 Considerações finais

Neste capítulo foram introduzidos conceitos básicos sobre trajetórias, possíveis análises que podem ser extraídas a partir de seus dados, bem como exemplos que utilizam tais análises.

Na seção sobre trabalhos relacionados foram apresentadas e em seguida discutidas as diferentes abordagens na literatura quanto ao tratamento dos dados de trajetórias e caminhos. Sendo dividido em dois principais abordagens, onde a primeira aborda a clusterização das mesmas e o segundo abordando a previsão.

4 FUNDAMENTOS DE MACHINE LEARNING

4.1 Introdução

O principal foco em ML está em responder questões acerca de como um algoritmo consegue "aprender"atividades específicas, tais como reconhecimento de certos tipos de vinho, reconhecimento de caracteres e vozes, separação de certos materiais, dentre diversos outros exemplos, e atrai cada vez mais a atenção de diversos pesquisadores no âmbito de resolver problemas cotidianos, fornecer dados para tomada de decisões, dentre outras finalidades, aplicado em diversas área (MELLO; PONTI, 2018). Tais algoritmos são organizados em dois principais tipos de aprendizado: supervisionado e o não supervisionado.

Algoritmos supervisionados são caracterizados por possuírem um conjunto de entradas já pré-classificado, muitas vezes fornecido por um *expert*, durante a etapa de treinamento, com a finalidade de convergir para o melhor classificador possível (MELLO; PONTI, 2018). Após esta fase de treinamento o algoritmo receberá novos dados e então classificará os mesmos, levando em consideração métricas de avaliação de desempenho definidas pelo desenvolvedor.

Para algoritmos não supervisionados as entradas do treinamento são fornecidas sem classificação e então o algoritmo procura identificar padrões ou semelhanças nestas entradas, uma vez identificada estas semelhanças, as mesmas serão agrupadas em conjuntos, tais conjuntos podem posteriormente ser analisados por especialistas ou em caso de *datasets* muito grandes, tal análise pode se inviável, sendo assim outras métricas serão analisadas para determinar o desempenho do algoritmo (MARSLAND, 2014).

4.1.1 Clusterização

Na etapa inicial lança-se mão de algoritmos de clusterização, cuja finalidade é agrupar objetos em subconjuntos, cada subconjunto sendo caracterizado como sendo um *cluster*, uma vez que objetos contidos em um mesmo *cluster* são semelhantes entre si, mas diferentes dos objetos pertencentes aos demais *clusters* (HAN; KAMBER; PEI, 2011). Os algoritmos de clusterização podem ser divididos em categorias, sendo detalhadas a seguir e que foram retiradas de (HAN; KAMBER; PEI, 2011):

Os métodos baseados em hierarquia, são caracterizados por possuírem duas classificações, aglomerativa ou divisiva. Na aglomerativa cada dado é iniciado como sendo um diferente grupo ou *cluster*, em seguida estes dados são mesclados até um grupo único (nível mais alto de hierarquia) ou até um determinado critério de parada, como por exemplo uma determinada quantidade de núcleos. Na classificação divisiva, todos os dados são iniciados compondo um único grupo e em seguida sendo divididos em grupos menores,

até cada objeto estar localizado em um *cluster* ou até determinado critério de parada. Os métodos hierárquicos podem ser baseados em distância, densidade e continuidade.

O método baseado em densidade parte do princípio de que o *cluster* irá agregando dados enquanto a densidade dos dados vizinhos exceda algum limite, como por exemplo quando o raio de um determinado *cluster* tiverem vizinhos que contenham um número mínimo de pontos, formando assim regiões densas no espaço de dados separadas por regiões dispersas.

Os métodos baseados em grade consistem no espaço de dados ser quantizado em um número finito de células que formam uma estrutura de grade, onde todas as operações de agrupamento serão realizadas nesta estrutura.

Para os algoritmos de clusterização serão utilizados neste trabalho o método de clusterização por densidade utilizando a distância de Fréchet. A distância de Fréchet consiste na medida de distâncias entre trajetórias que possuem formas contínuas, tais como formas curvilíneas e circulares, levando em consideração parâmetros como a localização e ordenação dos pontos ao longo destas trajetórias, medindo assim as similaridade entre curvas (AGARWAL et al., 2013). A definição utilizada na clusterização através da distância de Fréchet é baseada no trabalho de Buchin et al. (2008) a qual define que dada uma trajetória T com n vértices, a clusterização de uma sub-trajetória, a qual compreende o segmento entre vértices, é definida por $C_T(m, l, d)$, onde l representa o comprimento de pelo menos m sub-trajetórias não idênticas, a distância entre as subtrajetórias é de no mínimo d e pelo menos uma subtrajetória possui tamanho igual a l. Dado então um conjunto de m curvas $F = f_1, ..., f_m$, $f_i : [a_i, a'_i] \rightarrow \mathbb{R}^2$ a distância de Fréchet $\delta_F(f, g)$ pode ser definida como (BUCHIN et al., 2008):

$$\delta_F = \inf_{\substack{\alpha_1:[0,1]\rightarrow[a_i,a'_i] \\ \cdots \\ \alpha_m:[0,1]\rightarrow[a_m,a'_m]}} \max_{t\in[0,1]1\leq i,j\leq m} |f_i(\alpha_i(t)) - f_j(\alpha_j(t))| \tag{4.1}$$

Onde $\alpha_1, ..., \alpha_m$ são funções contínuas e decrescentes com $\alpha_i(0) = a_i$ e $\alpha_i(1) = a'_i$, i=0,...,m. Para o cálculo entre curvas arbitrárias, as mesmas são aproximadas por curvas poligonais.

O algoritmo baseado em densidade é o algoritmo *Density-based Spatial Slustering of Applications with Noise (DBSCAN)*, cujo principal objetivo é que para determinado *cluster*, a vizinhança considerando um determinado raio contém no mínimo um determinado número de pontos, onde a densidade na vizinhança tem que exceder este limiar para ser formado um *cluster* (CASSIANO, 2014).

4.1.2 Redes Neurais

Na etapa de predição de estados lança-se mão de algoritmos de Redes Neurais. Os quais são baseados em sistemas biológicos e cuja principal característica é encontrar representações matemáticas que se assemelhem ao processo de aprendizagem de um neurônio biológico como apresentado em Bishop (2006). Esta semelhança ao neurônio biológico também é encontrada em sua estrutura, uma vez que a topologia básica de uma rede neural é composta por um rede de neurônios conectados entre si, porém processo de aprendizagem é realizado através do ajuste de pesos, realizada durante o processo de treinamento, afim de que se obtenha as saídas desejadas da rede. Tais neurônios são agrupados em camadas, as quais dependem de quais neurônios estão conectados e a quais funções são aplicadas nas entradas, uma vez que tais camadas determinam o tipo de rede e executam operações equivalentes (KERFS, 2017).

Na etapa de previsão de trajetórias são utilizados algoritmos de redes neurais, mais especificamente algoritmos de RNN , sendo uma classe de algoritmos supervisionados, cuja principal característica é aprender padrões sequenciais ou variantes no tempo. Nesta classe de algoritmos tem-se a Memória de Longo e Curto Prazo (*Long Short Term Memory-LSTM*). Uma característica das RNN padrões consiste na utilização somente da memória de longo prazo ou de curto prazo, resultando em dificuldades durante a etapa de treinamento, já na LSTM este problema foi resolvido através das células de memórias, onde as mesmas atualizam e expõe seu conteúdo quando necessário, decidindo se é uma memória de curto ou longo prazo. Tal mecanismo é possível através dos *gates* que as células de memória possuem, que são *input gate, forget gate* e *output gate* , ou seja se for detectado que o conteúdo é importante o *forget gate* será fechado e transportará o conteúdo por ciclos, o que equivale a uma memória de longo prazo, por outro lado a memória pode considerar apagar o conteúdo da memória abrindo o *forget gate*, ambos os modos podem estar presentes simultaneamente em diferentes unidades de LSTM presentes em uma RNN (CHUNG et al., 2015).

Outro exemplo é o GRU, semelhante ao algoritmo de LSTM, onde sua a memória, também é exposta aos *gates* modulando o fluxo de informações, porém é totalmente exposta a cada ciclo, possuindo assim um único neurônio, os *gates* presentes na GRU são *reset gate* e *update gate*. O *update gate* irá controlar quanto da memória passada deve ser esquecido e quanto da memória atual será adicionada e sempre que uma informação for detectada anteriormente ou se a memória for considerada importante para o uso posterior, o *update gate* irá ser fechado e carregar esta memória através dos ciclos, já o *reset gate* é responsável por ignorar os estados ocultos anteriores sempre que for considerado necessário, ou seja auxilia na utilização eficiente da capacidade do modelo permitindo ser esquecida se uma informação não é mais necessária(CHUNG et al., 2015).

4.2 Trabalhos relacionados

Em Gudmundsson e Wolle (2014), Buchin et al. (2008), Aronov et al. (2006) e Buchin et al. (2018) é abordada a utilização da distância de Fréchet em algoritmos de clusterização, bem como o cálculo de tal distância e seu satisfatório desempenho ao ser aplicado em caminhos que possuem formas curvilíneas. Dentre os trabalhos citados merece destaque o apresentado em Gudmundsson e Wolle (2014), uma vez que aborda a utilização da clusterização de trajetórias de um jogo de futebol aplicando a distância Fréchet discreta, levando em consideração somente dados de posição, sem considerar a unidade de tempo.

Em Lee, Han e Whang (2007) é apresentado o algoritmo TRACLUS baseado na partição de caminhos e posterior agrupamento destes subcaminhos, utilizando o método baseado em densidade DBSCAN. A principal vantagem deste método está em descobrir subcaminhos comuns, uma vez que a diversidade de formas que as trajetórias de jogadores possuem, tal característica pode vir a fornecer *clusters* que possam representar de forma mais exatas as trajetórias dos mesmos.

Na etapa de previsão de trajetórias foram encontrados diversos artigos que apresentavam a utilização do algoritmo LSTM para previsão. Em Alahi et al. (2016), Choi, Song e Yoo (2019) e Xu, Piao e Gao (2018) utilizam LSTM para a previsão da trajetória, utilizando também parâmetros para medir a afinidade entre os pedestres e assim prever a trajetória de um pedestre com base na interação com os demais. Já em Zhang et al. (2019) o algoritmo LSTM é comparado com outros algoritmos de predição como *Echo State Network (ESN)* e filtro de Kalman, obtendo assim uma performance melhor que os demais. Outro método proposto para previsão de trajetórias é GRU, como abordado em (LEE; HAN; WHANG, 2007) o mesmo foi utilizado para reconhecimento de atividades em grupo.

4.3 Considerações finais

Este capítulo traz a definição da principal ferramenta utilizada neste trabalho, que são os algoritmos de ML, apresentando os algoritmos a serem utilizados que são os algoritmos de clusterização. Os trabalhos abordados na seção trabalhos relacionados, foram selecionados com base nos algoritmos utilizados como base para o desenvolvimento da proposta deste trabalho.

Os algoritmos para clusterização foram escolhidos com bases nos trabalhos onde os mesmos já foram aplicados no tratamento de trajetórias que apresentam formas mais complexas e apresentaram resultados satisfatórios.

5 FUNDAMENTOS DE MÁQUINA DE ESTADOS FINITA

5.1 Introdução

Máquina de estados finita (*Finite State Mahine - FSM*), são utilizadas com a finalidade de modelar comportamentos dos sistemas em diversas áreas, tais como circuitos, determinados tipos de *software*, protocolos de comunicação, dentre outros (LEE; YANNAKAKIS, 1996). A FSM é composta por estados, que são todas as possíveis situações que uma máquina de estados pode representar, e transições, que consistem nos eventos que determinam a mudança de um estado para o outro. A sequência destes estados e transições determinam uma cadeia, uma vez que a denominação de finita advém da característica dos estados serem finitos (WAGNER et al., 2006).

Formalmente uma FSM é composta por 5 elementos $M = \{I, O, S, \delta, \lambda\}$ (LEE; YANNAKAKIS, 1996):

- I são os conjuntos dos símbolos de entrada
- O são os conjuntos dos símbolos de saída
- S são os conjuntos de estados
- δ: S x $I \rightarrow S$ é a função da transição do estado
- λ: S x $I \rightarrow O$ é a função de saída

Seu funcionamento consiste em quando a FSM está no estado atual s em S e recebe uma entrada a de I, irá transicionar para o próximo estado especificado por $\delta(s,a)$ e produzir uma saída dada por $\lambda(s,a)$ (LEE; YANNAKAKIS, 1996).

Uma FSM pode ser representada através de dois modos, o primeiro consiste na matriz de transição, representada através de uma tabela, onde a linha (De) representam o estado atual e a coluna (Para) o próximo estado, onde a intersecção de uma linha com uma coluna é preenchida com a transição (WAGNER et al., 2006). Através da Tabela 1 pode-ser representada uma matriz de transição, que possui os estados representados por S1, S2 e S3 e as transições representadas por A1, A2 e A3.

O segundo modo de ser representada é através do diagrama de transição de estados, que consiste em uma representação gráfica. Nesta representação o círculo representa o estado e o arco, que se conecta aos círculos, representa a transição. Esta representação pode ser ilustrada na Figura 4, a qual representa a FSM ilustrada na Tabela 1, uma vez que ambas as representações ilustram a mesma FSM e são equivalentes. Porém em alguns

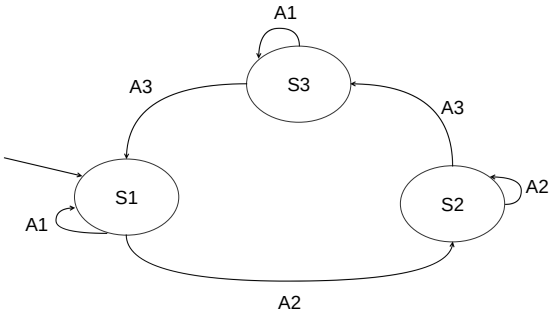
Tabela 1 – Matriz de transição

		Para		
		S1	**S2**	**S3**
De	**S1**	A1	A2	-
	S2	-	A2	A3
	S3	A3	-	A1

Fonte: Elaborada pelo autor

casos o diagrama de transição de estados é mais fácil de ser compreendido (WAGNER et al., 2006).

Figura 4 – Diagrama de transição de estados



Fonte: Elaborada pelo autor

A FSM possui dois tipos principais: máquina de Moore e máquina Mealy (LEE; YANNAKAKIS, 1996). A máquina de Moore é definida como sendo uma máquina sequencial, possuindo um número finito de estados, incluindo estado inicial, um número finito de entradas e saída. Sua principal característica está em as saídas serem determinadas pelo estado atual. Matematicamente pode ser representada por 6 elementos M=$\{X, I, O, F, x_0, G\}$ (HUANG, 2011):

- X são os conjuntos de estados
- I são os conjuntos das entradas
- O são os conjuntos de saída
- F: X x $I \rightarrow X$ é a função da transição do estado
- x_0 é o estado inicial
- G: $X \rightarrow O$ é a função de saída

A máquina de Mealy também é uma máquina sequencial, porém possui uma função de saída diferente da máquina de Moore. Uma vez que na máquina de Moore a saída depende somente do estado atual, na máquina de Mealy as funções de saída dependem do estado atual mas também da entrada. Podendo ser representada matematicamente por M=$\{X, I, O, F, x_0, G\}$ (HUANG, 2011):

- X são os conjuntos de estados
- I são os conjuntos das entradas
- O são os conjuntos de saída
- F: X x $I \rightarrow X$ é a função da transição do estado
- x_0 é o estado inicial
- G: X x $I \rightarrow O$ é a função de saída

5.2 Trabalhos relacionados

Diversos trabalhos podem ser encontrados na literatura utilizando FSM na área de software e hardware, podemos citar os trabalhos de Bergamaschi, Camposano e Michael (1991) onde aborda sua aplicação como descritor para a geração do circuito de *hardware*, em Ostanin (2017) utiliza para testar e assim encontrar defeitos em circuitos lógicos, em Das, Lerner e Seigle (2002) e Endsley, Lucas e Tilbury (2000) é abordado seu uso na verificação de algoritmos, dentre outros.

Encontramos também sua aplicação em outras áreas, tais como em Kempowsky, Subias e Aguilar-Martin (2006) que aborda a aplicação de FSM no monitoramento de processos em indústria, para que este monitoramento venha a auxiliar no processo de supervisão dos processos por parte do operador. Em Abella, Wright e Gorin (2004) é abordada a aplicação em diálogos, a qual irá ouvir todas as chamadas e fornecer então análises e diagnósticos, no âmbito da avaliação em tempo real e *business inteligence*. Em Kurt e Özgüner (2013) é a bordada a utilização em carros autônomos, sendo aplicada no controlador de alto nível do sistema.

5.3 Considerações finais

Neste capítulo foram introduzidos os principais conceitos de máquina de estados, ferramenta esta que constitui um dos principais objetivos específicos deste trabalho, sendo aplicada na classificação das jogadas.

Na seção trabalhos relacionados foram abordadas as principais pesquisas na área utilizando tal ferramenta. Onde pode ser destacados os trabalhos de Hong, Turk e Huang

(2000) e Verma e Dev (2009), apresentando uma abordagem muito semelhante a explorada neste trabalho, em que consiste na aplicação de FSM em algoritmos para classificação das jogadas, sendo apresentado com mais detalhes na Seção 6.2.

6 MATERIAIS E MÉTODOS

A proposta deste trabalho tem como objetivo geral a predição de padrões da interação entre múltiplos agentes em um ambiente dinâmico, sendo aplicada com a finalidade de realizar a classificação das jogadas consideradas com alto potencial de conversão em gol por parte do time adversário e utilizando algoritmos de predição, identificá-las durante o jogo.

Para o desenvolvimento desta proposta utilizou-se os dados do futebol de robôs simulado da sub-categoria 2D, caracterizado por ser um ambiente dinâmico, com uma grande interação entre os agentes e de difícil previsão, sendo assim o cenário ideal para aplicação desta proposta.

Para alcançar este objetivo geral foram definidos 3 objetivos específicos e que estão ilustrados no diagrama geral representado através da Figura 5, que serão detalhados nas próximas seções, que são eles a preparação do *dataset* a ser utilizado, classificação das jogadas de interesse e por fim a previsão das mesmas.

Figura 5 – Diagrama esquemático da arquitetura geral proposta

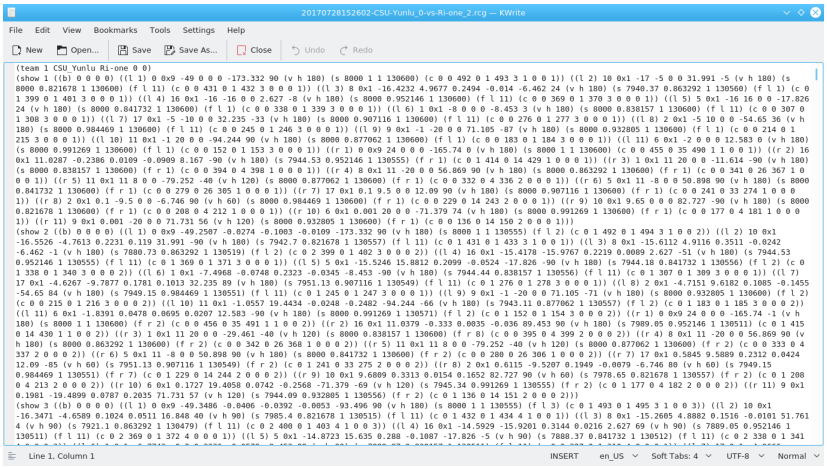
Fonte: Elaborada pelo autor

6.1 Elaboração do Dataset

Para geração do *dataset*, base de dados utilizadas como entrada dos algoritmos, foram utilizados os *logs* gerados a partir dos jogos da Liga de Simlulação 2D, disponibilizados pela *RoboCup* em RoboCup (2019b), a ferramenta *Logplayer* será de grande utilidade para

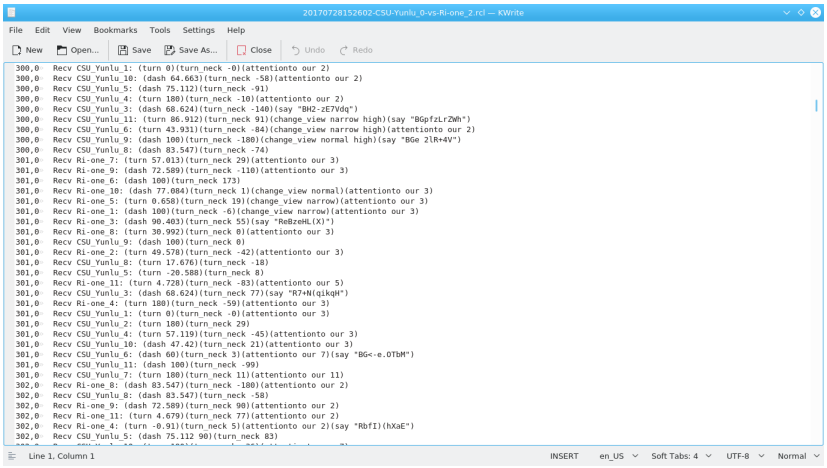
análise e obtenção de parâmetros visuais para a seleção dos dados através de um *expert*. Tais *logs*, gerados pelo *soccerserver*, necessitam de um pré-tratamento, pois em sua forma crua são ordenados de maneira de difícil interpretação, como ilustrado nas Figuras 6 e 7, e com uma grande quantidade de dados que não serão utilizados nesta abordagem, tais como o comando *dash*, que consiste na energia que o jogador consome para acelerar, gerando assim a necessidade de um pré tratamento para os mesmos. Foi então implementado um algoritmo para filtrar estes dados, ilustrado na Figura 8 e detalhado seu desenvolvimento nesta seção.

Figura 6 – Log do arquivo no formato .rcg gerado



Fonte: Elaborada pelo autor

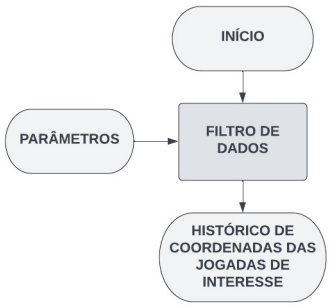
Figura 7 – Log do arquivo no formato .rcl gerado



Fonte: Elaborada pelo autor

O passo inicial foi interpretar os dados contidos nos logs, onde através dos trabalhos de Reis (2003) e o manual da Robocup RoboCup (2019a), foram utilizados o arquivo do tipo *.rcg*, contendo as informações de *playmode*, ou seja os comandos de arbitragem que o juiz comunica durante o jogo. Inicialmente, foram gerados os arquivos, com os

Figura 8 – Diagrama esquemático do algoritmo para geração do *Dataset*



Fonte: Elaborada pelo autor

seguintes dados: ciclo, nome do time, número do jogador, posição x e y da bola e do jogador, velocidade x e y da bola e do jogador e o estado de *playmode*, utilizando como base o programa de Yarn (2018).

A etapa seguinte consiste em determinar quais as jogadas serão selecionadas para a etapa da classificação, sendo definidas 4 tipos de eventos de interesse a serem classificadas como efetivamente resultar em gol e potencialmente resultar em gols. Estes eventos de interesse consistem em: uma jogada que resulta em gol, em que goleiro pega a bola, que resultam escanteio e resultam em tiro de meta. Sendo as mesmas identificadas através da arbitragem do juiz durante o jogo, que são dados que contém as informações *goal, goalie_ catch_ ball, corner_ kick, goal_ kick*, seguidas da informação do time a que se destina a jogada. Além da determinação destas jogadas é necessário também a determinação da janela de tempo anterior a esta arbitragem, a qual será considerada como sendo de 150 ciclos, sendo obtida através da análise de tais jogadas por um *expert*.

Após a seleção das jogadas de interesse foi definido que para o objetivo desta etapa seriam gerados somente os caminhos dos jogadores do time analisado mais a bola, porém como o jogo considera os dois períodos de jogo, temos jogadas acontecendo em ambos os lados do campo de futebol. Sendo necessário um último filtro para que os caminhos gerados sejam orientados somente para um dos lados do campo, o lado direito.

Através das etapas descritas anteriormente o algoritmo para filtrar o dados, ilustrado a partir da Figura 8, terá como entrada o *dataset* geral, composto pelo arquivo *.rcg*, conforme ilustrado na Figura 6, e em decorrência das etapas descritas anteriormente, irá gerar como saída o *dataset*, Figura 9, que contém o histórico de caminhos das jogadas de interesse. Neste dataset cada linha é composta por coordenadas x e y, na Figura 9 a primeira informação é a coordenada x valor 48.1257 e a segunda informação a coordenada y valor -31.944, ambas correspondendo ao ciclo 150, em seguida o terceiro dado corresponde a coordenada x valor 46.6268 e o quarto dado a coordenada y valor -30.0421 do ciclo 149, sendo que esta primeira linha corresponde ao jogador 1, ou seja irá compor o caminho percorrido pelo jogador 1, a segunda linha contem as mesmas informações do jogador 2.

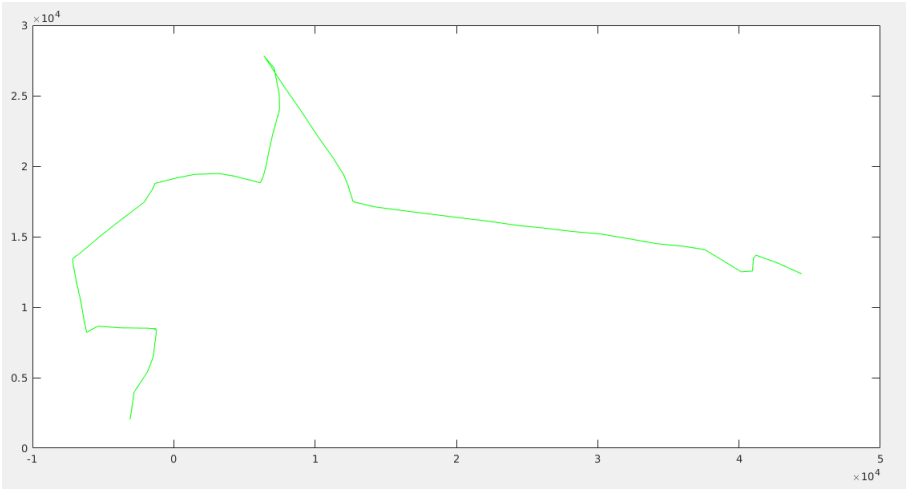
Onde cada linha corresponde as coordenadas percorridas por cada jogador no caminho que o mesmo percorreu dada a janela de tempo máxima de 150 ciclos, dada a seqência dos 11 jogadores mais a bola, daquela jogada de interesse.

Figura 9 – Coordenadas das jogadas de interesse do time Ri-One

Fonte: Elaborada pelo autor

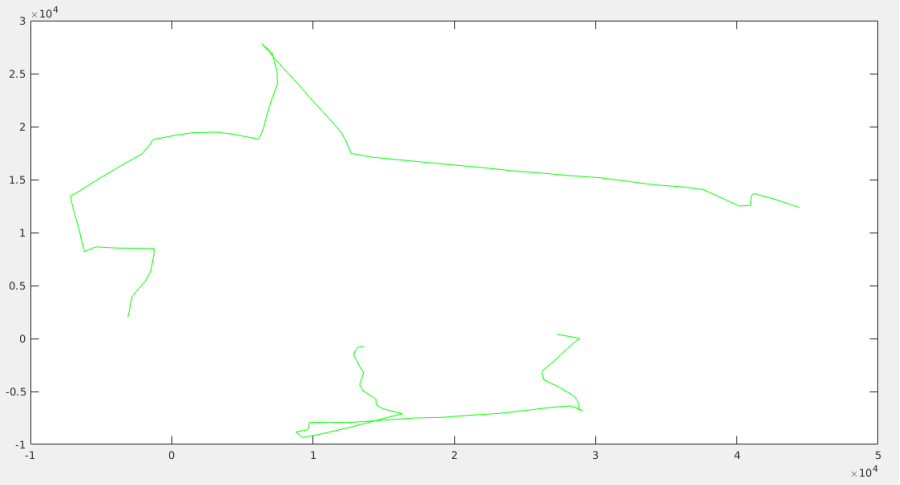
Uma vez que a definição de caminho, conforme já introduzido, é definida como a sequência de coordenadas sem a dependência temporal, conforme descrito no parágrafo anterior a composição do dataset, é ilustrado na Figura10 o caminho percorrido por um jogador, composto pelas coordonadas do mesmo dada a janela de tempo máxima 150 ciclos de uma jogada de interesse. Na Figura 11 ilustra o caminho de 2 jogadores diferentes em jogadas de interesse também diferentes.

Figura 10 – Caminho de um jogador no campo



Fonte: Elaborada pelo autor

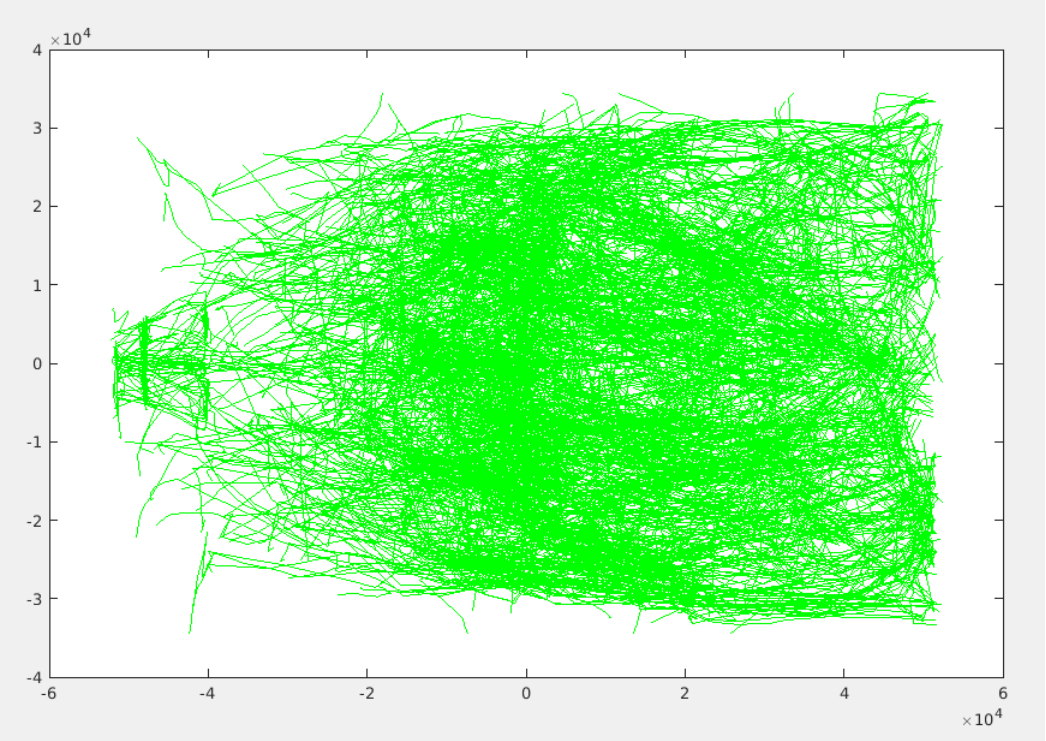
Figura 11 – Caminho de dois jogadores no campo



Fonte: Elaborada pelo autor

Na Figura 12 foram gerados todos os caminhos dos jogadores mais a bola do time Ri-One das jogadas de interesse dos anos de 2019 e 2018. O código fonte utilizado para geração deste *dataset* está disponível em Barbosa (2022).

Figura 12 – Coordenadas das jogadas de interesse do time Ri-One



Fonte: Elaborada pelo autor

6.2 Classificação das jogadas

Nesta etapa será utilizado o *dataset* que contém o histórico de caminhos das jogadas de interesse, gerado durante a etapa anterior, onde as informações necessárias para a clusterização serão somente as coordenadas x e y de cada jogador e da bola, a fim de gerar a classificação das jogadas de interesse. Para gerar esta classificação foi dividido em 3 principais etapas, que são a clusterização dos caminhos, geração das cadeias representadas por máquinas de estados e o algoritmo classificador de jogadas. O código fonte desta implementação está disponível em Barbosa (2022).

6.2.1 Clusterização dos caminhos

Para clusterização dos caminhos será utilizado o algoritmo de clusterização por densidades, o qual utiliza o cálculo de distâncias de Fréchet, baseado no algoritmo proposto por Buchin et al. (2008). Ambos descrito no Capítulo 4 e através deste serão obtidas os principais caminhos dos jogadores e da bola. Na Figura 13 pode ser visualizado o diagrama esquemático desta etapa da clusterização dos caminhos.

Figura 13 – Diagrama esquemático do algoritmo de clusterização de caminhos

Fonte: Elaborada pelo autor

O algoritmo que utiliza a distância de Fréchet, foi escolhido por clusterizar caminhos contínuos, sendo mais apropriado para formas curvilíneas se comparado a outras distâncias, tais como Hausdorf conforme apresentado em Buchin et al. (2008), bem como verificado através da literatura sua aplicação em caminhos no contexto de futebol simulado de robôs, como em Gudmundsson e Wolle (2014). Para implementação deste algoritmo foi basedado no programa disponibilizado em Meulemans (2019) e abordado em Buchin et al. (2013), onde foram realizadas alterações para a proposta deste trabalho, a seguir será descrito o algoritmo utilizado.

A entrada do algoritmo de clusterização de caminhos consiste no histórico de coordenadas das jogadas de interesse, as quais foram obtidas a partir dos caminhos gerados na etapa de elaboração do dataset descrita na Seção 6.1, sendo então gerados os principais caminhos, ou seja, os clusters. De posse destes caminhos a etapa inicial consiste em

normalizá-los. Em seguida é cálculado a distâncias entre todos os caminhos utilizando a distância de Fréchet.

Esta proposta de Buchin et al. (2013) apresenta uma otimização no cálculo da distância de Fréchet. A definição clássica desta distância utiliza a metáfora do cão e seu dono, onde ambos estão andando em caminhos curvilíneos e conectados por uma guia, ambos podem variar sua velocidade porém nunca caminhar para trás, onde a distância de Fréchet consiste na menor distância da guia que une o homem e seu cão durante todo o trajeto dos seus respectivos caminhos. Sendo assim varia-se tamanhos de guias para verificar se a distância entre ambos é viável e a partir uma escolha inteligente da guia, uma delas irá obter a distância de Fréchet otmizada. A otimização deste algoritmo consiste em possuir uma guia retrátil sendo aumentada e reduzida, reduzindo assim o tempo de execução do algoritmo.

A partir do trabalho de Buchin et al. (2013), a distância de Fréchet para esta aplicação é definida a partir da teoria de *distance terrain*, uma generalização do *free space diagram* tipicamente utilizada na teoria clássica da distância de Fréchet. Onde na *distance terrain* tem-se a *acrophobia function*, cuja definição é baseada através do caminho π:[0.1] $\rightarrow \mathbb{R}$ bimonótono, para (s,t) ϵ $\mathbb{R}$, temos $\Pi(s,t)$ é o conjunto de todas os caminhos bimonótonos da origem até (s,t). A função que representa a menor altura para atingir um ponto (s,t) a partir da origem de um caminho monótono, consiste na *acrophobia function* definida como:

$$\tilde{T}(s,t) = \inf_{\pi \epsilon \Pi(s,t)} \max_{\lambda \epsilon [0,1]} T(\pi(\lambda)) \tag{6.1}$$

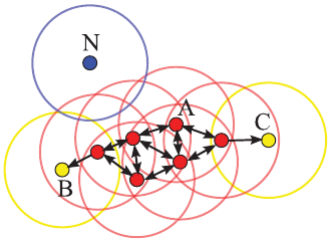
A partir destas definições temos a célula da *distance terrain*, tem-se então a definição de $L_{\mathrm{i,j}}$ como sendo o lado esquerdo da célula e $B_{\mathrm{i,j}}$ o lado inferior da célula, e por consequência define-se a *acrophobia function* ao longo do lado correspondente como sendo $\tilde{L}^*_{\mathrm{i,j}}$e $\tilde{B}^*_{\mathrm{i,j}}$. Sendo então a distância de Fréchet obtida através da seguinte equação :

$$d_{\mathrm{F}}(P,Q) = max\{\delta(P(m),Q(n)), min\{\tilde{L}^*_{\mathrm{m\text{-}1,n\text{-}1}}, \tilde{B}^*_{\mathrm{m\text{-}1,n\text{-}1}}\}\} \tag{6.2}$$

Através do cálculo da distância de Fréchet é realizada a clusterização por densidade utilizando o algoritmo DBSCAN. Este tipo de clusterização por densidade necessita da definição de dois parâmetros para o agrupamento dos clustres, que são: o número mínimo de vizinhos *minPts* e o raio ε. Onde objetos com um número igual ou superior a quantidade de vizinhos *minPts* dentro do raio ε delimitado são considerados como possíveis *core points*, todos os vizinhos dentro deste raio são considerados dentro deste cluster e são denominados de diretamente densamente atingíveis. Caso estes pontos sejam considerados como *core points*, atingindo o *minPts* dentro do raio ε, seus vizinhos serão incluídos, sendo densamente atingíveis conforme Figura 14 sendo os pontos em vermelho, porém

caso não seja considerado um *core point*por não atingirem o mínimo número de vizinhos, são denominados de *border point*, uma vez que são considerados densamente conectados pertencentes a um cluster, conforme Figura 14 sendo os pontos em amarelo. Os pontos que não são densamente atingíveis de qualquer outro ponto são considerados ruídos e não pertencem a nenhum cluster, conforme Figura 14 sendo os pontos em azul, conforme descrito em Schubert et al. (2017), código fonte desta implementação está disponível em Barbosa (2022).

Figura 14 – Ilustração da clusterização utilizando o algoritmo DBSCAN



Fonte: (SCHUBERT et al., 2017)

6.2.2 Geração das sequências das jogadas

Após realizada a clusterização dos caminhos, a etapa seguinte consiste em gerar a sequência de todas as jogadas de interesse descritas nas subceções anteriores. Esta sequência de jogadas será implementada através da representação das jogadas por meio da FSM abordada no Capítulo 5, sendo representada através de uma cadeia que consiste em uma sequência de estados.

De posse dos principais clusters gerados pelo algoritmo de clusterização e as coordenadas, provenientes do *dataset* do histórico de coordenadas das jogadas de interesse, será identificado qual cluster mais próximo de determinada coordenada de cada jogador e da bola, identificação esta que será realizada através da medida de distância entre a coordenada até o núcleo mais próximo através da distância de Fréchet.

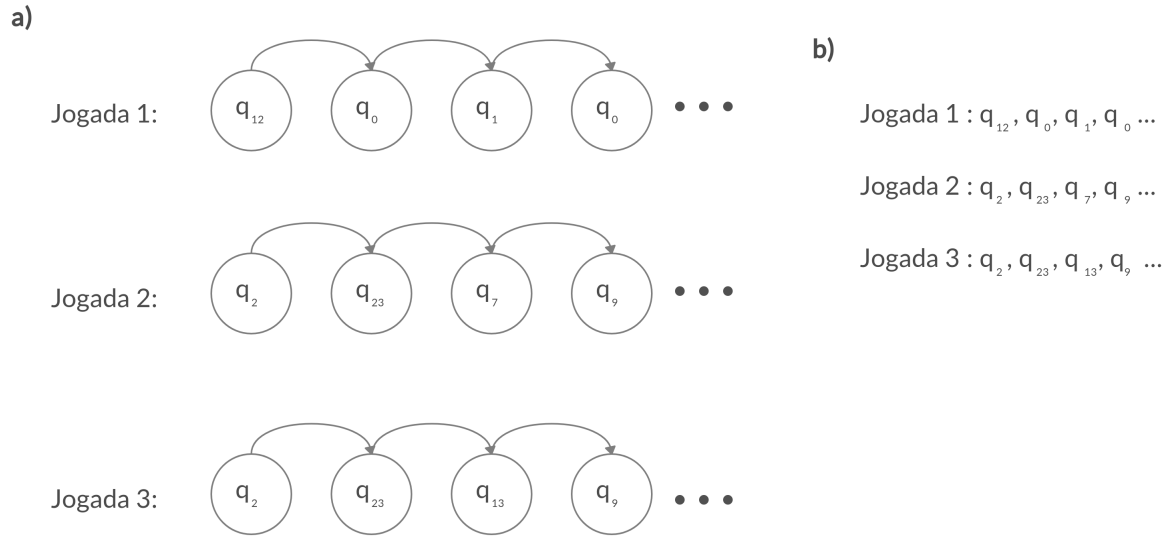
Após a identificação de cada ciclo teremos um conjunto de caminhos de todos os jogadores e a bola, a partir destas informações será obtido um vetor que irá representar os estados q_i composto por 12 posições, onde as primeiras 11 posições serão os caminhos identificados naquele ciclo dos jogadores seguida da posição do caminho da bola. A seguir será apresentado um exemplo do estado q_i, sendo para fins de exemplificação considerado que foram identificados 20 caminhos principais, sendo classificados como sendo de 0 a 19 e sendo representado somente o primeiro estado q_0.

$$q_0 = \begin{bmatrix} 0 & 18 & 15 & 0 & 1 & \dots & 7 & 11 & 3 & 3 \end{bmatrix}^T$$

Como o algoritmo da etapa seguinte, clusterização de máquinas de estados, possui uma implementação cujo tempo de processamento aumenta com o aumento da quantidade de estados acima de 256 estados, uma vez que com o aumento do tamanho das matrizes há a necessidade de um processamento maior que o disponível, foram implementadas generalizações na geração dos estados, para que fossem gerados uma quantidade de estados menores e assim reduzir o custo computacional. A primeira simplificação foi realizada ao considerar que dado um estado não importa o caminho de cada jogador e sim a composição dos caminhos. A segunda simplificação é de que mesmo se um dos caminhos for diferente o estado será considerado como sendo o mesmo.

A sequência da máquina de estados é composta pelos estados identificados anteriormente, onde o máximo tamanho da cadeia será de 150, número máximo de ciclos anteriores a jogada de interesse ocorrer. De posse da representação dos estados e transições, podemos definir então as jogadas como sendo um sequência de estados e transições, formando cadeias das jogadas de interesse. Podendo ser representadas como o exemplo abaixo, onde cada linha representará a sequência de uma jogada de interesse.

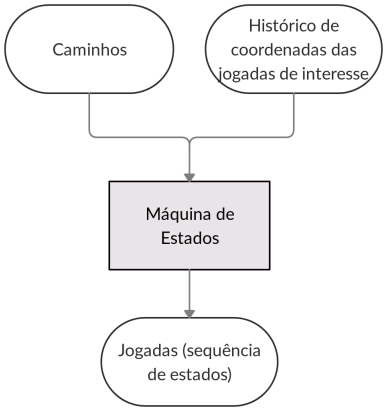
Figura 15 – Representação das cadeias das jogadas de interesse



(a)Diagrama de transição de estados das jogadas de interesse; (b)Representação da sequência de estados das jogadas de interesse. **Fonte:** Elaborada pelo autor

O algoritmo possui então como entrada os principais caminhos gerados através da clusterização e as coordenadas de cada jogador e da bola de todas as jogadas de interesse. E como saída é gerada as sequências de todas as jogadas de interesse de todos os jogos. Tal implementação é ilustrada através da Figura 16, que contém o diagrama esquemático do algoritmo. O código fonte desta implementação está disponível em Barbosa (2022).

Figura 16 – Diagrama esquemático do algoritmo gerador das cadeias das jogadas de interesse

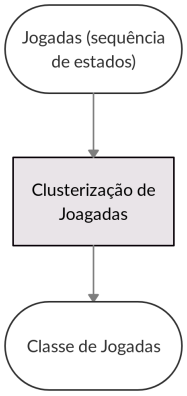


Fonte: Elaborada pelo autor

6.2.3 Classificação de jogadas

De posse das cadeias de todas as jogadas, é realizada a etapa de classificação das jogadas, uma vez que tem-se uma grande quantidade de jogadas obtidas na etapa de geração das cadeias, as mesmas deverão ser agrupadas e em seguida classificadas. O algoritmo a ser utilizado é ilustrado através da Figura 17.

Figura 17 – Diagrama esquemático do algoritmo classificador de jogadas



Fonte: Elaborada pelo autor

A entrada do algoritmo de classificação consiste nas sequências de todas as jogadas de interesse, obtidas na etapa de geração de cadeias descrita na Subseção 6.2.2, e a saída serão as jogadas agrupadas e classificadas, ou seja as classes das jogadas de interesse.

Para a clusterização de máquinas de estados foi utilizado o algoritmo *Identity* abordado por Girgis, James e Luczak (2021), o mesmo é utilizado para clusterização de sequências de DNA. Este algoritmo foi escolhido com base na característica de as sequências de DNA as quais são utilizadas para clusterização se assemelham aos estados

encontrados.

O Algoritmo *Identity* apresentado por Girgis, James e Luczak (2021) consiste na clusterização baseada no cálculo da *Identity score* o qual é definido como sendo a porcentagem dos nuceotídios idênticos entre duas sequências genômicas alinhadas. Este algoritmo para clusterização de sequências genônimicas abordado por Girgis, James e Luczak (2022) foi utilizado como base para a implementação do algoritmo para clusterização da sequência de máquinas de estados.

Uma vez que o dataset de entrada é composto por uma quantidade menor que 10000 sequências, as mesmas serão mutadas, descasadas, deletadas, inseridas, substituídas ou duplicadas unitariamente ou através de blocos. Esta etapa é necessária para evitar a utilização de algoritmos para alinhamento das sequências, uma vez que este algoritmo para longas sequêcias pode ser impraticável. Este tipo de mutação faz-se necessário para que o dataset possua o tamanho necessário para a etapa de treinamento, onde ao final desta etapa, serão gerados 2 datasets , um para treinamento e outro para teste.

A etapa seguinte consiste em calcular as estatísticas de cada par de histograma para as sequências. O cálculo destas estatísticas são as seguintes:Manhattan, Euclidean, x^2, Chebyshev, Hamming, Minkowski, Cosine, Correlation, Bray Curtis, Squared chord, Hellinger, Conditional KL divergence, K divergence, Jeffrey divergence, Jensen-Shannon divergence, Revised relative entropy, Intersection, Kulczynski 1, Kulczynski 2, Covariance, Harmonic mean, Similarity ratio, Markov, SimMM and $D_2{}^S$. Sendo calculadas estas estatísticas para as sequências de entrada e em seguidas normalizadas.

Através destas estatísticas são selecionadas 3 estatísticas para compor a matriz **F** de *features*, onde as mesmas são selecionadas com base no erro quadrático médio. Esta etapa é realizada no treinamento, onde é realizada a combinação das matrizes para calcular a melhor composição que diminui o erro quadrático médio durante a etapa de treinamento do algoritmo de predição. Para o algoritmo de predição é utilizado o algoritmo *General Linear Model* (GLM), sua forma geral é descrita a partir da equação:

$$y = F\omega \tag{6.3}$$

Onde y é o valor da *Identity score* e **F** é a matriz de *features*, o coeficiente omega é obtido a partir da pseudo-inversa, conforme equação seguinte.

$$\omega = (F^{\mathrm{T}}F)^{\text{-}1}F^{\mathrm{T}}y \tag{6.4}$$

De posse do coeficiente da GLM obtemos assim a equação 6.5 para previsão, onde $\hat{y}$ representa a predição do valor da *Identity score* dado uma sequência de pares.

$$\hat{y} = \mathbf{F}\omega \tag{6.5}$$

Através da equação 6.5 será realizado o treinamento da rede e assim através dos parâmetros do erro médio quadrático (*Mean Average Error - MAE*) e do erro médio absoluto (*Mean Squared Error - MSE*) são utilizados como métrica para a etapa de treinamento. É nesta fase que o cálculo da *Identity* será realizado, este valor será calculado através da média ponderada das estatísticas selecionadas, os valores acima do limiar definido pelo usuário serão consideradas e em seguidas estes valores calculados serão utilizados para agrupar as sequências. O código fonte desta implementação está disponível em Barbosa (2022).

6.3 Preditor de jogadas

Na etapa de previsão de jogadas através da posição atual de cada jogador, será gerada a probabilidade da jogada em andamento pertencer a determinada classe, definida previamente na etapa de classificação, sendo então implementada a rede neural LSTM, a ser descrita neste capítulo.

Conforme apresentado por Yu et al. (2019) áreas em que posuem dados sequenciais, como textos, audios e outros, a utilização de RNN é dominante, onde a principal característica é que conexões deste tipo de rede serem cíclicas, possuindo a capacidade de atualizar o estado atual com base no estado passado e o dado atual. Sendo assim pode-se aplicar a mesma neste caso de estudo, uma vez que faz-se necessária a previsão da probabilidade do estado atual pertencer a determinada classe de jogada com base no estado atual e passado.

Conforme descrito por Chung et al. (2015) a arquitetura da rede LSTM é composta por uma célula de memória, onde carrega a memória da rede, e o *input gate*, *forget gate* e *output gate* controlam a exposição e as mudanças dos conteúdos da memória. Esta memória é composta pela entrada atual e a memória prévia modulada pelos *forget gate* e *input gate*, os quais controlam quanto do conteúdo deverá ser memorizado e quanto do antigo conteúdo deverá ser esquecido. O *output gate* controla a quantidade que o conteudo da memória será exposto.

Visando a implementação do algoritmo de LSTM foi desenvolvido um algoritmo em python utilizando a biblioteca tensorflow, tal implementação utiliza a implementação padrão de um algotimo LSTM, sendo utilizando a plataforma *Google Colab* para desenvolvimento deste algoritmo.

Para os dados recebidos foram separados em treinamento e teste na proporção de 80% para teste e 20% para treino. Em seguida foi utilizada a camada *Embedding* a qual irá ralizar a transformação dos estados discretos em um vetor espacial denso de um tamanho fixo, é nesta etapa que define-se a utilização do *padding* com zeros no dataset, para que

todas as sequências do dataset de entrada possuam o tamanho fixo de 150 estados. Em seguida foi utilizada a camada da rede neural LSTM, sendo fixado o tamanho da camada escondida em 100. Posteriormente foi utilizada a camada *Dense* cuja função é conectar densamente a rede neural, cujos parâmetros utilizados foram a quantidade na camada de saída da rede, definida pelo número de classes de cada time, e definida a função de ativação como sendo a sigmoide.

A última camada utilizada é a *compile* para configuração do treinamento da rede, utilizando os seguintes parâmetros como a função de perda, cuja função é quantificar a quantidade que o modelo deveria seguir para minimizá-la durante o treinamento, sendo utilizada a função *binary crossentropy* a qual irá calcular a cross entropia de perda entre a verdadeira classificação e a predita , otimizador *Adam*, que é um método gradiente descendente baseado na estimação adaptativa de primeira e segunda ordem, sendo computacionamente eficiente e que requer pouca memória e a métrica escolhida, para avaliação é utilizada a métrica de acurácia, estes parâmetros foram obtidos e verificados apartir da documentação da biblioteca Keras Keras (2022). O código fonte desta implementação está disponível em Barbosa (2022).

6.4 Considerações Finais

Neste capítulo foi abordada a proposta de trabalho desenvolvida ao longo deste mestrado, que pode ser dividida em 3 principais objetivos. Sendo que o primeiro consiste na preparação do *dataset*, onde foram descritas a obtenção dos dados, a partir dos jogos de futebol de robô simulado da categoria 2D dos anos de 2019 e 2018, e em seguida o tratamento destes dados para a criação do *dataset* que contém o histórico de coordenadas das jogadas de interesse.

A fase de classificação das jogadas consiste inicialmente na clusterização dos caminhos, posterior identificação das cadeias de todas as jogadas, utilizando FSM, e em seguida classificadas as principais jogadas de interesse.

Após a etapa de classificação foram utilizados os algoritmos de previsão, com a finalidade de prever durante o jogo a probabilidade de determinada jogada em andamento pertencer a uma das classes, encontradas na etapa anterior.

7 RESULTADOS EXPERIMENTAIS

Após a descrição dos algoritmos implementados para este estudo, neste capítudo serão abordados os principais resultados referente a implementação.

7.1 Elaboração do Dataset

Conforme descrito na seção 6.1 para geração do dataset tem-se uma gama muito grande de times para análise, sendo selecionados 3 times para esta aplicação, ITAdroids, Ri-One e Helios. O Time ITAdroids foi escolhido por ser um time nacional, o time Ri-One foi escolhido por ter apresentado boas colocações porém próximo aos anos analisados não obteve colocações próximas aos ganhadores, o time Helios é um time tradicional nesta categoria e apresentou bons resultados nos anos analisados.

Foram selecionados inicialmente para análise 3 anos, 2017, 2018 e 2019. No decorrer do experimento na etapa de clusterização das máquinas de estados, descrita na Seção 6.2.3 foi observado que o algoritmo para clusterização de sequências que apresentavam um número superior a 256 estados, o consumo de memória RAM aumentava proporcionalmente ao número de estados e sequências geradas, para times como o Helios, que apresentavam uma maior quantidade de estados, não possuía memória suficiente durante a etapa de treinamento da rede para execução do programa. Dada esta condição para uma análise mais uniforme dos times foram consideradas as análises dos jogos de 2 anos, 2019 e 2018, com isto tornando possível o processamento dos dados para todos os times.

Ao executar o algoritmo para o perído dos anos de 2018 e 2019 e obter as jogadas de interesse destes 3 times, pode-se observar que para o time Ri-One foram gerados um total de 1344 caminhos, para o time do ITAdroids 1896 caminhos e para o time do Helios 2940 caminhos. Sendo assim pode-se inferir que o time do Helios possui um número superior de jogadas de interesse, podendo ser uma indicação de que o mesmo seja um time mais ofensivo que os demais e obtenha melhores resultados nas competições.

7.2 Classificação das jogadas

7.2.1 Clusterização de caminhos

De posse do dataset descrito na Seção 7.1, foi implementado o algoritmo DBSCAN utilizando a distância de Fréchet descrito na Seção 6.2.1. Como a implementação de um algoritmo que utiliza a clusterização por densidade são definidos 2 parâmetros: O raio ε e mínima quantidade de pontos da vizinhança *minPts*. Para a determinação dos valores destes dois parâmetros baseou-se nas definições abordadas no trabalho de Schubert et al. (2017). Para o parâmetro *minPts*, considerou-se inicialmente como sendo igual a duas

Tabela 2 – Valores de ruído e tamanho máximo do maior cluster para diferentes valores de ε e $minPts$=4

ε	**Ruído**	**Tamanho Máximo Cluster**
8.5	1137	155
9.0	1012	319
9.5	866	566
10.0	692	706
10.5	562	909
11.0	381	1189
11.5	271	1458

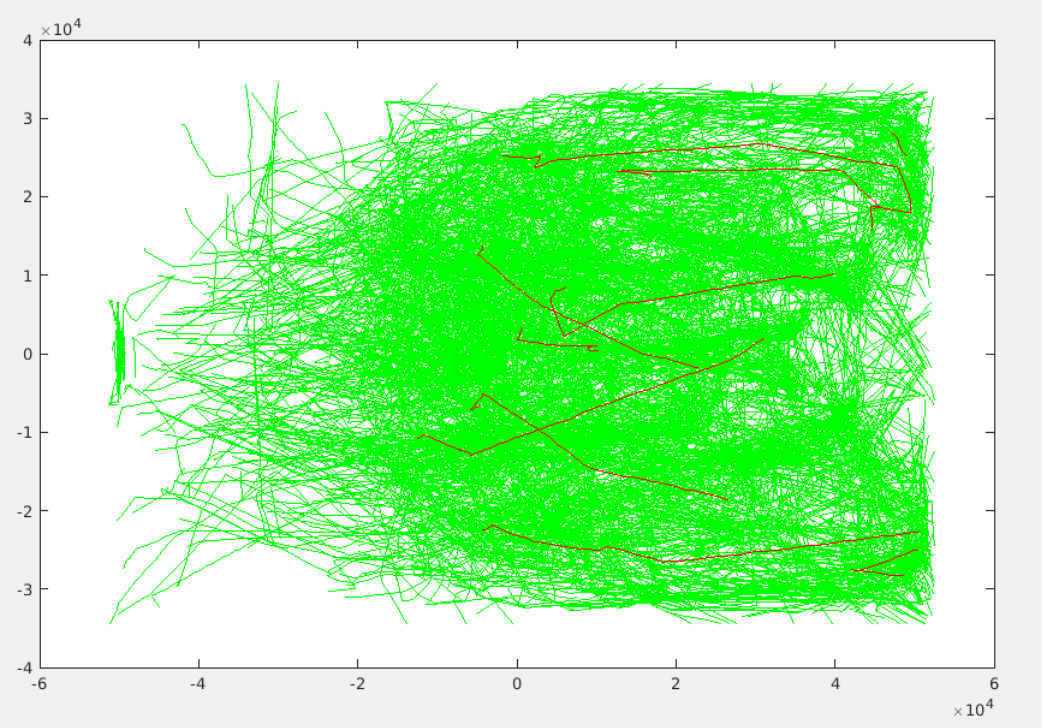
Fonte: Elaborada pelo autor

vezes o tamanho da dimensionalidade do dataset, como o dataset a ser utilizado possui duas dimensões o mesmo foi fixado como $minPts$=4. Para a definição do parâmetro de ε, será utilizada a abordagem apresentada para identificação de resultados de clusters degenerados, onde aborda as seguintes métricas que são a quantidade de ruído e o tamanho máximo do maior cluster como indicativo de um resultado não ótimos. O ruído, conforme detalhado na Seção 6.2.1 que são caminhos que ao final da clusterização não pertencem a nenhum cluster, foram obtidos da clusterização estejam entre 1% e 30% e o maior cluster formado possua entre 20% e 50% dos caminhos. Sendo então utilizados os critérios do valor de ruído e o tamanho máximo do maior cluster formado para selecionar os valores de $minPts$ e ε e assim obter a clusterização.

A partir destas métricas foram variados os valores de ε de forma a atender estes critérios, foram realizados experimentos que tiveram as seguintes saídas para o time do ITAdroids, onde a quantidade de caminhos totais, ou seja o dataset foi de 1896 caminhos. Fixando-se $minPts$=4 obteve-se os seguintes resultados apresentados na Tabela 2

Observando-se assim que com aumento de ε tem-se uma diminuição do ruído, porém um aumento do tamanho do maior cluster. Ao observar os resultados o único valor que satisfaz a ambas condições descritas no parágrafo anterior é o valor de ε=10.5. Porém com este resultado o algoritmo seguinte para gerar a quantidade de estados e sequências resultou em um valor do número de estados igual a 540, uma vez que este número de estados é maior que 256 o tempo de execução do algoritmo utilizado para clusterização de estados aumenta bem como a quantidade de memória utilizada na geração dos dados. Optou-se então por otimizar este valor do número de estados gerados no algortimo seguinte através da alteração do valor de $minPts$ para 6 e 8. Através das novas simulações destes dois valores de $minPts$ o melhor resultado obtido foi para $minPts$=8 e ε=11.0, uma vez que os valores de ruído=649 e máximo cluster = 914 estando o mais próximo possível de satisfazer a ambos os critérios de escolha definidos. Resultando em um total de 11 clusters, podendo ser vizualizado através da Figura 18, onde os caminhos são representados em verde os clusters em vermelho.

Figura 18 – Coordenadas das jogadas de interesse do time ITAdroids com clusters



Fonte: Elaborada pelo autor

Uma vez que para o valor de *minPts*=4 obtivemos resultados semelhantes para o time Ri-One e Helios, a fim de reduzir o número de estados e otimizar a execução do algoritmo seguintes foram também variados os valores de *minPts* para 6 e 8.

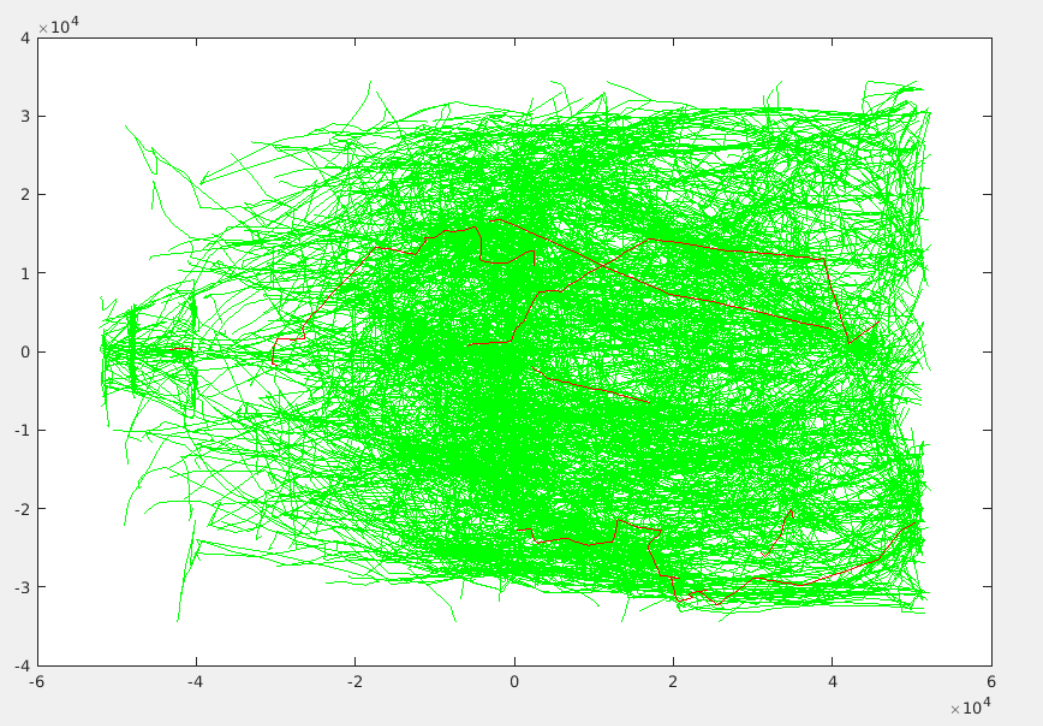
Para o time Ri-One possuindo um total de 1344 caminhos, o resultado que atendia o mais próximo das condições de ruído e tamanho do maior cluster foi *minPts*=8 e ε=10.5, obtendo os valores de ruído=452 e máximo cluster=652. Resultando em um total de 7 clusters, podendo ser vizualizado através da Figura 19, onde os caminhos são representados em verde os clusters em vermelho.

Conforme descrito anteriormente para o time Helios, qual possui uma quantidade total de caminhos de 2940, o melhor resultado obtido para um menor processamento do algoritmo seguinte foi com *minPts*=8 e ε=11.0, obtendo os valores de ruído=868 e máximo cluster=1674. Resultando em um total de 6 clusters, conforme Figura 20, onde os caminhos são representados em verde os clusters em vermelho.

Através dos resultados obtidos pode-se inferir que o time ITAdroids possuem clusters mais uniformemnte distribuídos pelo campo de futebol, podendo ser uma indicação de que as jogadas apresentam um a maior variação, ou seja são mais distribuídas e apresentam diferentes comportamentos.

O time Ri-One apresentou clusters em regiões mais concentradas, ao centro e lateral inferior, podendo ser observados dois padrões de principais caminhos.

Figura 19 – Coordenadas das jogadas de interesse do time Ri-One com clusters



Fonte: Elaborada pelo autor

Para time Helios pode-se observar que os principais caminhos se concentram na parte inferior do campo, podendo ser uma indicação de padrão de jogadas mais centralizado na região inferior do campo.

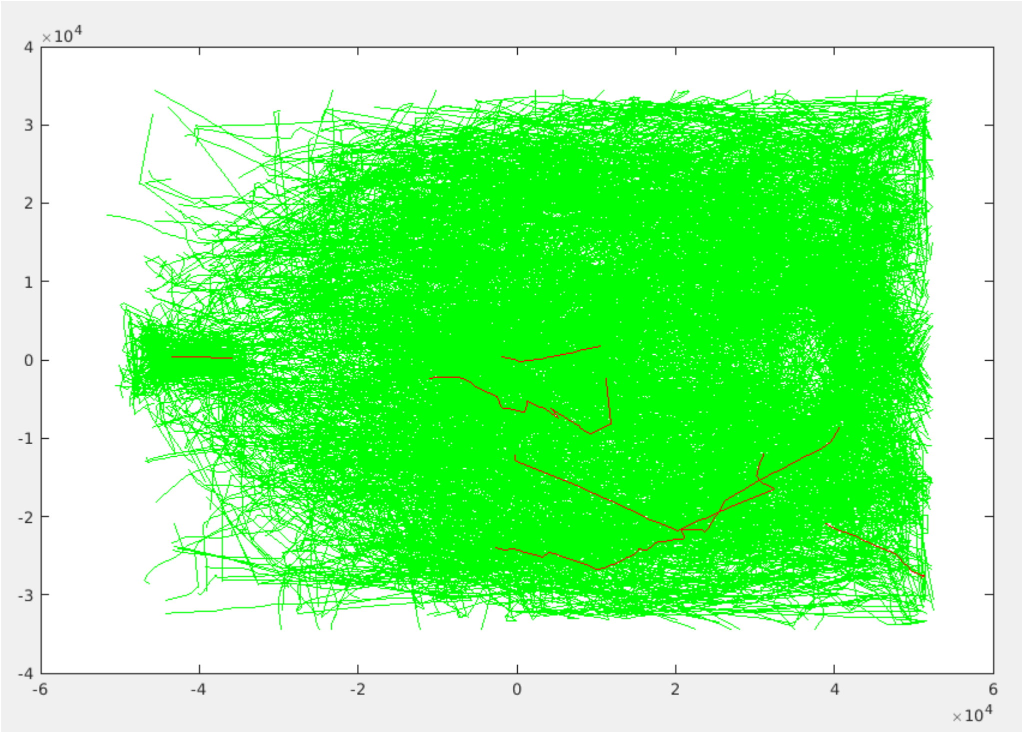
7.2.2 Geração das sequências das jogadas

A etapa seguinte consiste no algoritmo para gerar as sequências e os estados das jogadas de interesse. De posse dos *dataset* e dos clusters gerados no programa anterior gerou-se as sequências para o time ITAdroids, Ri-One e Helios, conforme o algoritmo descrito na Seção 6.2.2.

Nas Figuras 22, 23 e 24 apresentam o começo das sequências e os respectivos estados de cada um dos times através da Figura 25. Visando obter o menor número de estados possíveis, conforme descrito anteriormente uma vez que o algortimo seguinte exigia um esforço computacional proporcional a quantidade de estados e sequências a serem processadas, optou-se pela generalização dos caso onde somente a diferença de um dos caminhos que compõe o estado é considerado um mesmo estado.

Nesta etapa para o time ITAdroids foram gerados uma quantidade de 157 sequências, Figura 22, com um total de 320 estados, Figura 25 . Para o time Ri-One foram gerados 111 sequências, Figura 23 com um total de 155 estados, Figura 25 . E para o time Helios foram gerados uma quantidade de 244 sequências, Figura 24 com um total de 273 estados,

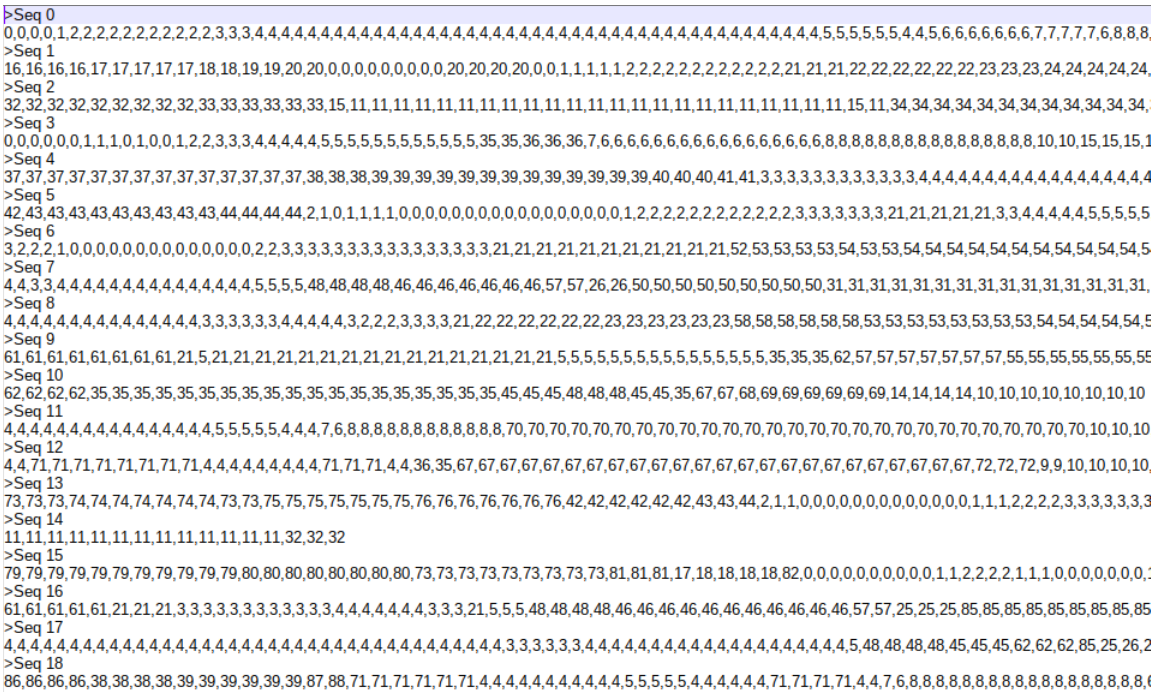
Figura 20 – Coordenadas das jogadas de interesse do time Helios com clusters



Fonte: Elaborada pelo autor

Figura 25.

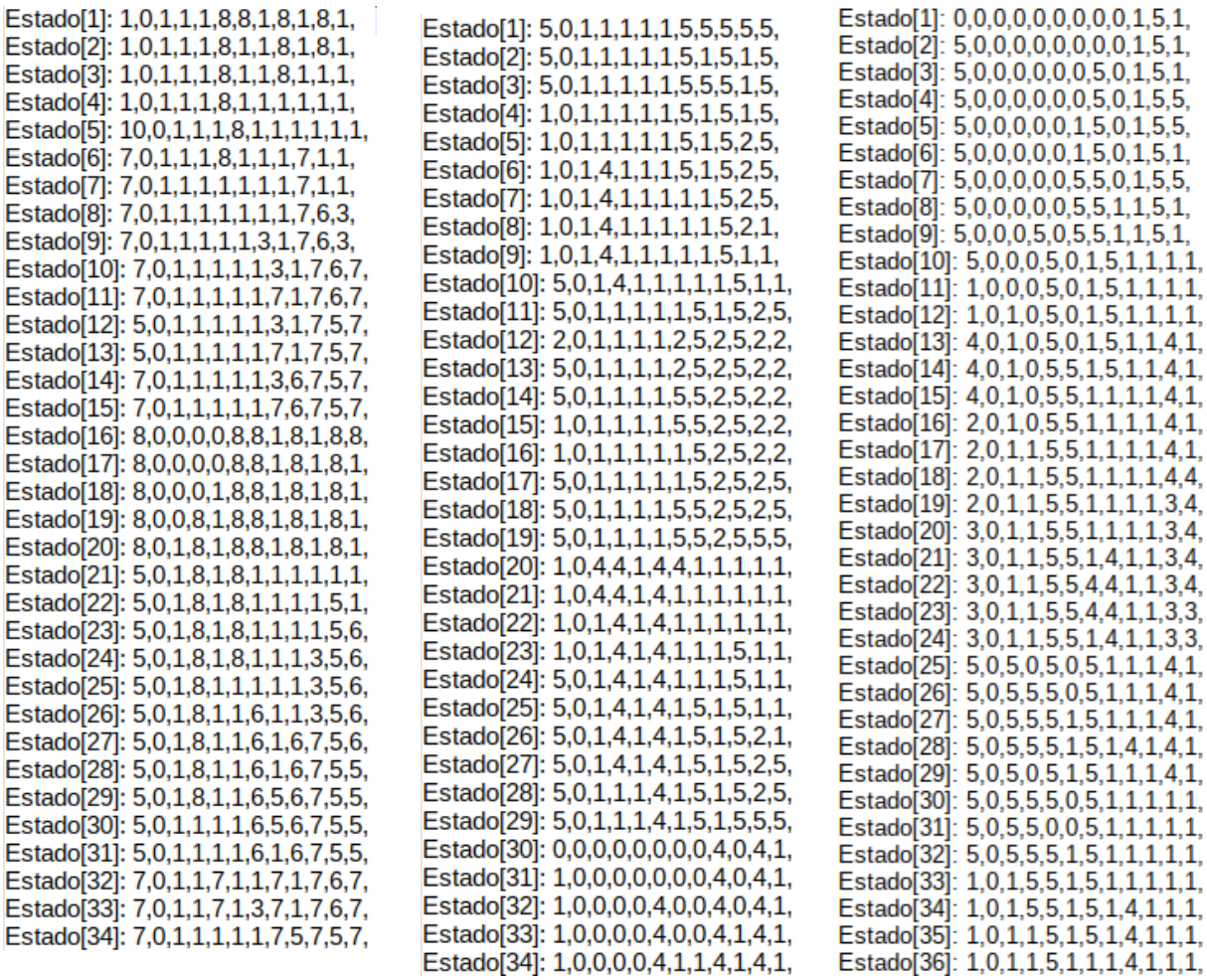
Figura 22 – Sequências geradas time ITAdroids



```
>Seq 0
0,0,0,0,1,2,2,2,2,2,2,2,2,2,2,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,5,5,5,5,5,5,4,4,5,6,6,6,6,6,6,7,7,7,7,7,6,8,8,8,
>Seq 1
16,16,16,16,17,17,17,17,17,18,18,19,19,20,20,0,0,0,0,0,0,0,0,0,20,20,20,20,0,0,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,21,21,21,22,22,22,22,22,22,23,23,23,24,24,24,24,24,
>Seq 2
32,32,32,32,32,32,32,32,32,33,33,33,33,33,33,15,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,15,11,34,34,34,34,34,34,34,34,34,34,34,34,
>Seq 3
0,0,0,0,0,0,1,1,1,0,1,0,0,1,2,2,3,3,3,4,4,4,4,4,5,5,5,5,5,5,5,5,5,5,5,5,35,35,36,36,36,7,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,10,10,15,15,15,
>Seq 4
37,37,37,37,37,37,37,37,37,37,37,37,37,37,38,38,38,39,39,39,39,39,39,39,39,39,39,39,39,40,40,40,41,41,3,3,3,3,3,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,
>Seq 5
42,43,43,43,43,43,43,43,43,43,44,44,44,44,2,1,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,21,21,21,21,21,3,3,4,4,4,4,4,5,5,5,5,5
>Seq 6
3,2,2,2,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,21,21,21,21,21,21,21,21,21,21,52,53,53,53,53,54,53,53,54,54,54,54,54,54,54,54,54,54,54,5
>Seq 7
4,4,3,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,5,5,5,5,48,48,48,48,46,46,46,46,46,46,57,57,26,26,50,50,50,50,50,50,50,50,31,31,31,31,31,31,31,31,31,31,31,31,31,31,
>Seq 8
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,3,3,3,3,3,3,4,4,4,4,4,3,2,2,2,3,3,3,3,21,22,22,22,22,22,22,23,23,23,23,23,23,58,58,58,58,58,58,53,53,53,53,53,53,53,53,54,54,54,54,54,5
>Seq 9
61,61,61,61,61,61,61,61,21,5,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,35,35,35,62,57,57,57,57,57,57,55,55,55,55,55,55,55
>Seq 10
62,62,62,62,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,45,45,45,48,48,48,45,45,35,67,67,68,69,69,69,69,69,69,14,14,14,14,10,10,10,10,10,10,10,10
>Seq 11
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,5,5,5,5,5,4,4,4,7,6,8,8,8,8,8,8,8,8,8,8,8,8,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,70,10,10,10
>Seq 12
4,4,71,71,71,71,71,71,71,71,4,4,4,4,4,4,4,4,4,71,71,71,4,4,36,35,67,67,67,67,67,67,67,67,67,67,67,67,67,67,67,67,67,67,67,67,67,67,67,72,72,72,9,9,10,10,10,10,
>Seq 13
73,73,73,74,74,74,74,74,74,74,73,73,75,75,75,75,75,75,75,76,76,76,76,76,76,76,42,42,42,42,42,42,43,43,44,2,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,2,2,2,2,3,3,3,3,3,3,3
>Seq 14
11,11,11,11,11,11,11,11,11,11,11,11,11,32,32,32
>Seq 15
79,79,79,79,79,79,79,79,79,79,79,80,80,80,80,80,80,80,73,73,73,73,73,73,73,73,81,81,81,17,18,18,18,18,82,0,0,0,0,0,0,0,0,0,0,1,1,2,2,2,2,1,1,1,0,0,0,0,0,0,0,
>Seq 16
61,61,61,61,61,21,21,21,3,3,3,3,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,3,3,3,21,5,5,5,48,48,48,48,46,46,46,46,46,46,46,46,46,46,57,57,25,25,25,85,85,85,85,85,85,85,85,85
>Seq 17
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,5,48,48,48,48,45,45,45,62,62,62,85,25,26,2
>Seq 18
86,86,86,86,38,38,38,38,39,39,39,39,39,39,87,88,71,71,71,71,71,71,4,4,4,4,4,4,4,4,4,4,4,4,5,5,5,5,5,4,4,4,4,4,4,71,71,71,71,4,4,7,6,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,
```

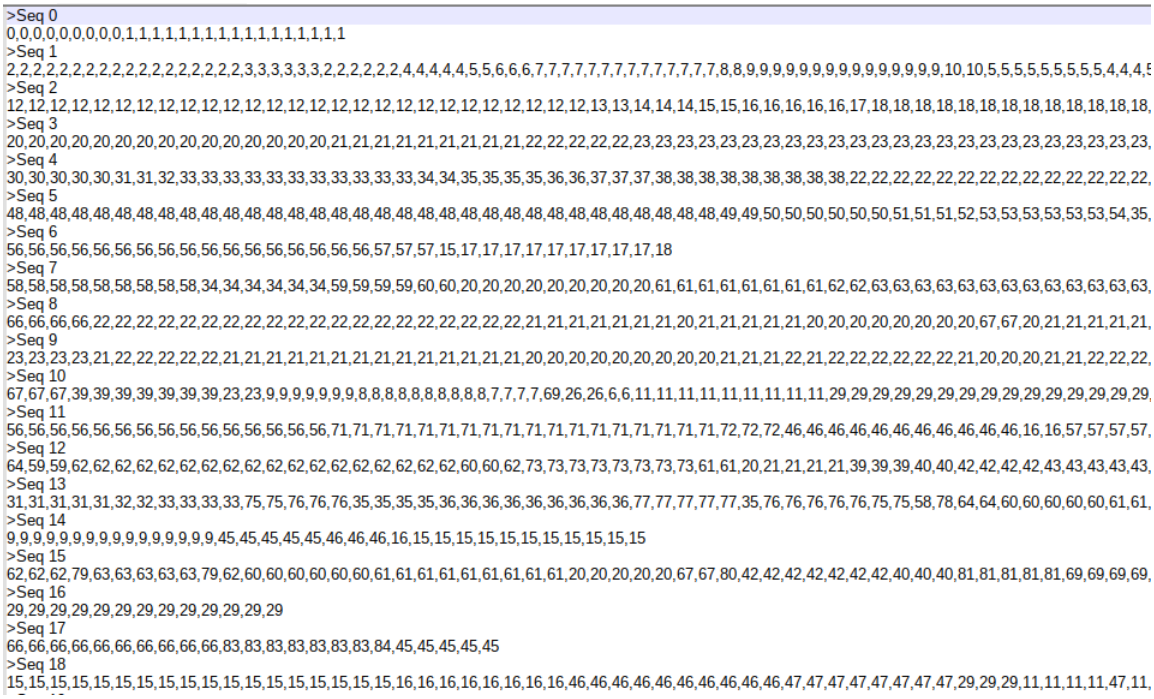
Fonte: Elaborada pelo autor

Figura 21 – Estados encontrados dos times:
(a) ITAdroids (b)Ri-One (c)Helios



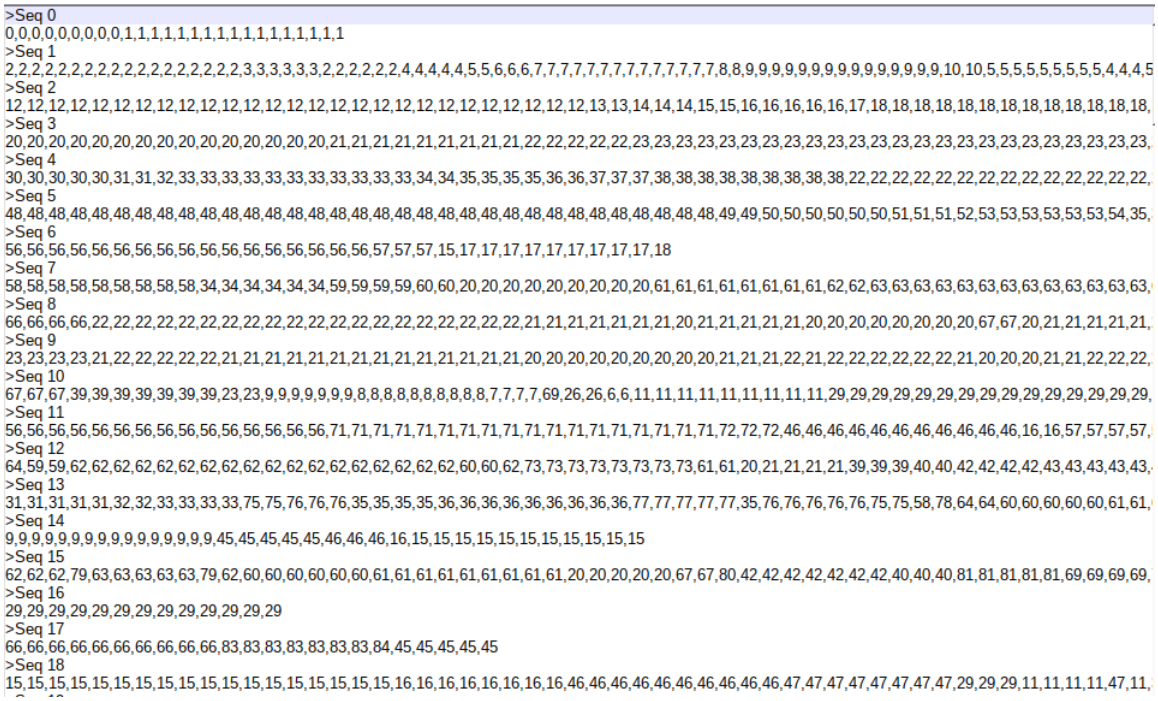
Fonte: Elaborada pelo autor

Figura 23 – Sequências geradas time Ri-One



Fonte: Elaborada pelo autor

Figura 24 – Sequências geradas time Helios



Fonte: Elaborada pelo autor

7.2.3 Classificação de jogadas

De posse das sequências dos estados anteriores, ilustradas nas Figuras 22,23 e 24 para obtenção das classes de jogadas lança-se mão dos algoritmos utilizados para clusterização de sequências de genes. Uma vez que as sequências geradas não possuem métricas de distâncias espaciais entre si, sendo necessário encontrar outras métricas e realizar a composição para obtenção de uma pontuação para agrupamento das mesmas. Onde através do algotimo *Identity* abordado em Girgis, James e Luczak (2021) e Girgis, James e Luczak (2022) o mesmo foi adaptado para o cenário da clusterização de sequências de estados de um jogo de futebol. Baseado no cálculo da *Identity score* para então realizar o agrupamento destes estados.

Os parâmetros necessários a ser definido pelo usuário foram t, c, b e v, o parâmetro t irá determinar o limiar do cálculo da identidade para determinar se uma sequência será ou não membro de um cluster, ou seja, se a pontuação for acima deste limiar a mesma será incluída na etapa de clusterização. O parâmetro c, corresponde a quantidade de cores ou hyperthreads a ser utilizada para procesamento paralelo, este valor foi definido como sendo menor que a quantidade disponível fisicamente do computador que é de 24. O valor mínimo necessário para treinamento e teste somam um total de 10000 sequências de entrada torna-se necessário então definir os parâmentros de b tamanho do bloco e v tamanho do bloco lido, estes parâmetro são utilizado durante a mutação do dataset, conforme descrito na Seção 6.2.3. Uma vez que valor de b e v são definidos como v=4*b e o menor valor

de b indicado é definido como 1000, estes parâmetros foram definidos como b e v=4000. Para este algortimo foi observado que quanto menor o valor de t maior é o número de sequências classificadas, portanto foram escolhidos os 3 menores valores de t, sendo t=0.1 t=0.2 e t=0.3, como critério de avaliação da clusterização foi utilizado os valores de MAE erro absoluto médio e MSE erro quadrático médio porém segundo abordado em Willmott e Matsuura (2005), a melhor métrica para comparação de performance é o MAE, sendo então utilizado definição do melhor resultado juntamente com a quantidade de clusters gerados.

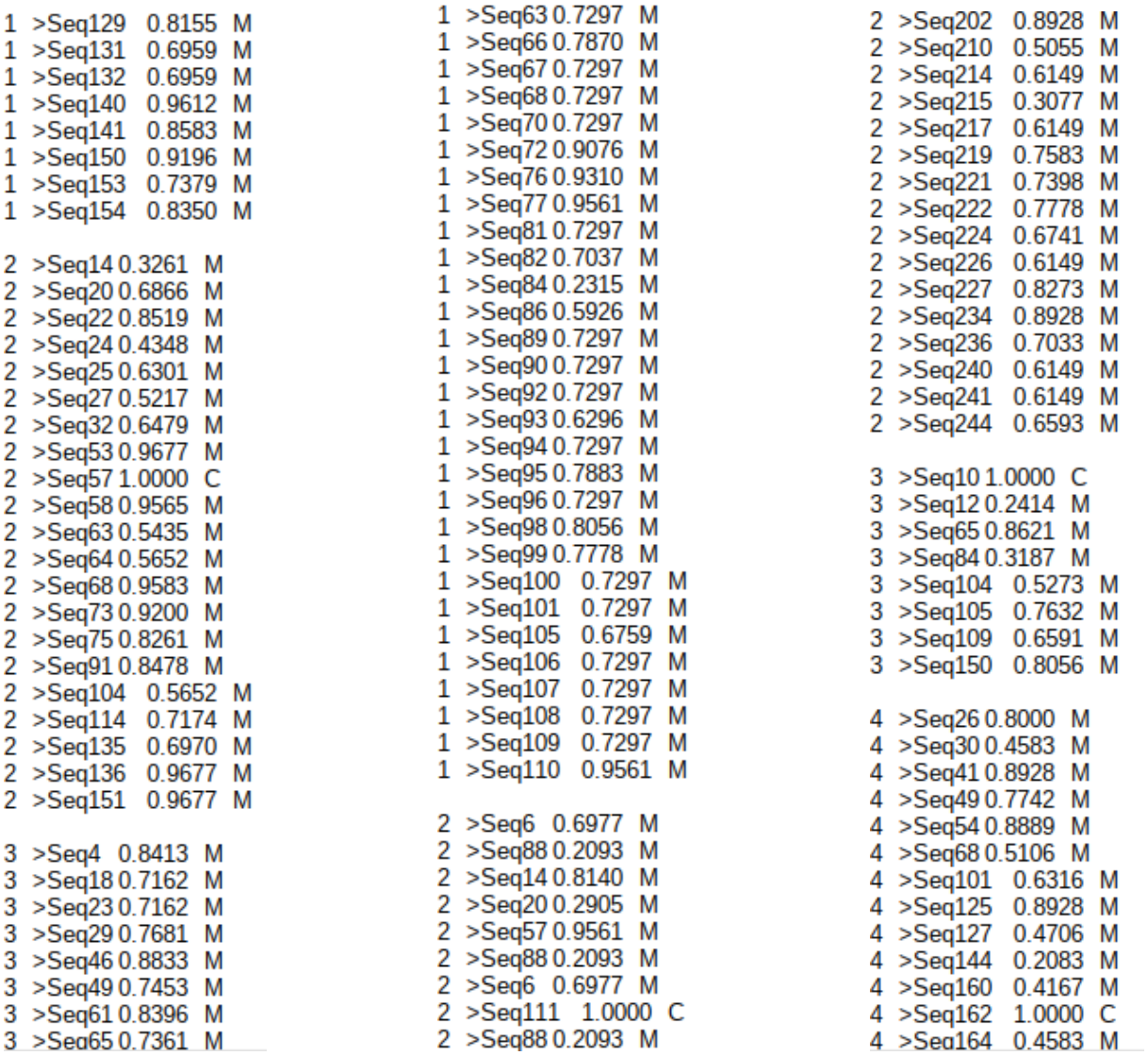
Para o time Ri-One, para verificação do resultado foi simulado o parâmetro t=0.1 , t=0.2, t=0.3 . Para t=0.1 MAE: 0.0007867 MSE: 1.31981e-05, para t=0.2 MAE:0.00092114 MSE:1.39546e-05, t=0.3 MAE: 0.000838068 MSE: 1.28661e-05 sendo assim a melhor performance de MAE foi para t=0.1, porém para este resultado obteve-se somente 2 classes com sequências classificadas e 42 sequências não classificadas, uma vez que foram gerados poucos clusters, optou-se por considerar a segunda melhor métrica de MSE sendo t=0.2, para esta classificação foram classificadas 68 sequências, gerando um total de 9 clases, as demais 43 sequências as quais não foram classificadas, ou seja não geraram um valor para Identity acima do limiar setado para t=0.2, foram agrupadas em um décimo cluster.

Para o time ITAdroids, para verificação do resultado foi simulado o parâmetro t=0.1 e t=0.2. Para t=0.1 MAE:0.000492133 MSE:5.70754e-06, para t=0.2 MAE:0.000421045 MSE:4.257e-06 sendo assim a melhor performance baseado no maior número de clusters gerados e na melhor performance de MAE foi t=0.2. Sendo classificadas 107 sequências, gerando um total de 10 clusters, as demais 50 sequências não classificadas, foram agrupadas em um décimo primeiro cluster.

Para o time do Helios , para verificação do resultado foi simulado o parâmetro t=0.1 , t=0.2 t=0.3 . Para t=0.1 MAE:0.000293 MSE:3.16415e-6 , para t=0.2 MAE:0.000787062 MSE:5.53434e-06 , t=0.3 MAE: 0.0003065 MSE:3.49277e-06 e para t=0.3 MAE:0.0007870 MSE:5.53434e-06, sendo assim a melhor performance é apresentada para t=0.1. Sendo classificadas 155 sequências, gerando um total de 4 clusters, as demais 87 sequências não classificadas, foram agrupadas em um quinto cluster.

Durante o treinamento deste algoritmo foi observado que quanto menor o valor de t mais sequências são classificadas e mais uniforme são as classes obtidas, favorecendo a etapa seguinte que consiste na predição de jogadas. Na Figura 25 encontra-se os clusters gerados nesta etapa, onde a esquerda da sequência encontra-se o cluster a qual ela pertence e a direita o valor da métrica *Identity score* utilizada para o agrupamento destas classes.

Figura 25 – Clusters das sequência de estados geradas para os times:
(a)ITAdroids (b)Ri-One (c)Helios



Fonte: Elaborada pelo autor

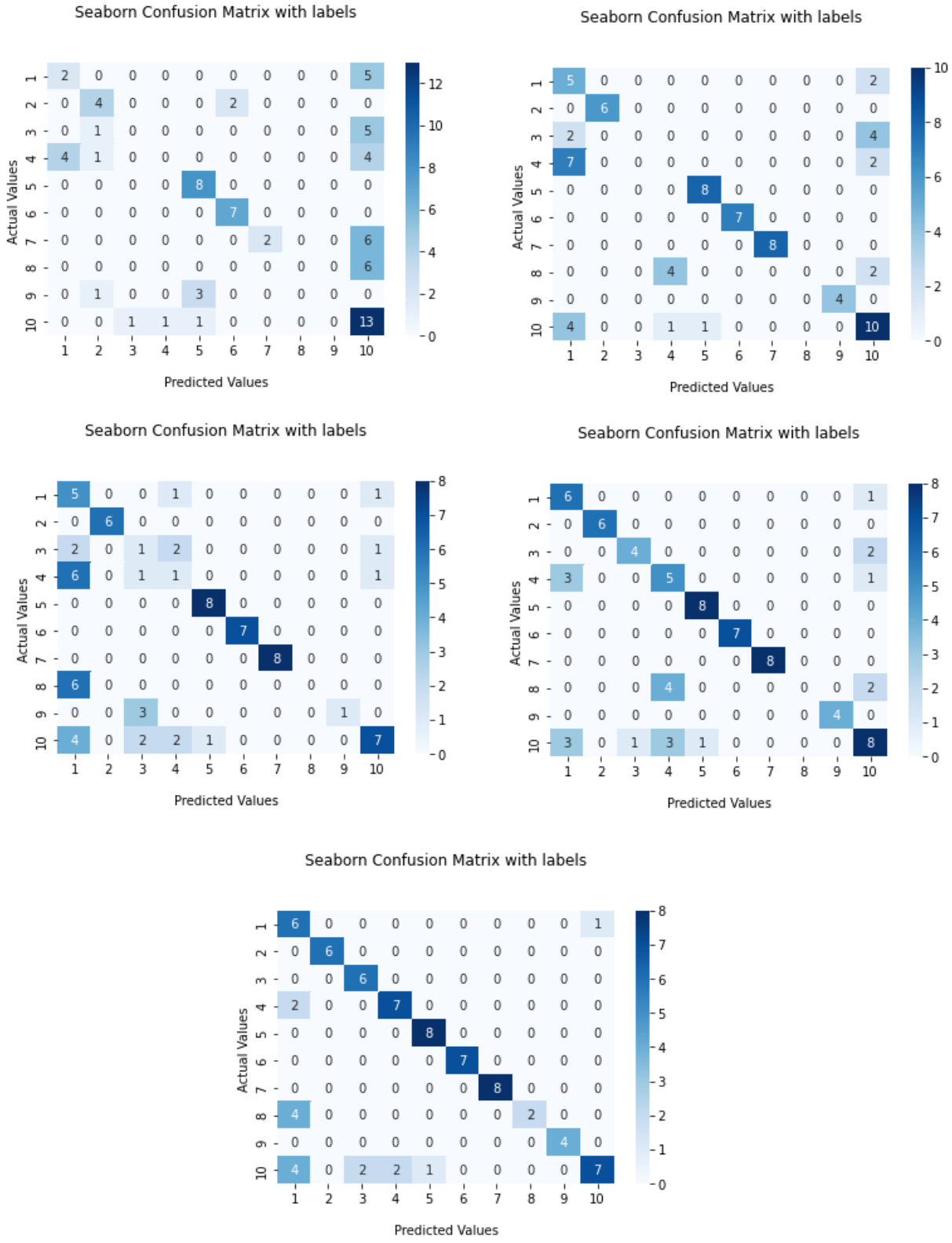
7.3 Preditor de jogadas

Na etapa de previsão de jogadas lança-se mão do algoritmo de LSTM para previsão das jogadas a partir da classificação das sequências obtidas. Foi desenvolvido o algoritmo padrão LSTM utilizando a biblioteca *Tensorfow* foi necessário aplicar o *padding* na sequência de entrada, ou seja completar as sequências de tamanho menor do que 150 com determinado bit de forma que todas as sequências de entrada possuam o mesmo tamanho, para utilização desta função é necessário que o *padding* seja feito utilizando o número zero, uma vez que o zero era utilizado como um estado foi necessário somar 1 no valor dos estados e o zero então utilizado como *padding* para este algoritmo.

As variáveis utilizadas para treinar o time Ri-One são: número de estados da

sequências 157, o número de classes 10, seguida do número máximo de elementos de uma sequência, que é uniforme de 150 para todas as entradas. Para o treinamento destas sequências, obteve-se uma acurácia global de 52,17%, realizando um balanceamento neste dataset, ou seja, completando com as respectivas sequências existentes as classes desbalanceadas, obteve-se uma melhora no resultado para 79.22%. Para uma análise mais detalhada, utilizando este dataset balanceado, foram geradas as matrizes de confusão a partir dos dados de teste das entradas do treinamento considerando a distribuição nas seguintes janelas de entrada 30, 60, 90, 120, 150, conforme a Figura 26 através destas métricas observa-se que com a evolução das janelas a rede converge para as corretas classificações, ou seja nas janelas iniciais temos uma maior taxa de erro na classificação dos estados, conforme a janela vai aumentando obtém-se a convergência dos valores para a correta classificação. Sendo também gerados os valores de precisão, sensibilidade e F1 de cada uma das janelas, conforme Tabelas 3, 4, 5, 6, 7.

Figura 26 – Matriz de confusão do time Ri-One nas janelas:
(a)30 (b)60 (c)90 (d)120 (e)150



Fonte: Elaborada pelo autor

Tabela 3 – Métricas de avaliação time Ri-One para janela de entrada 30

Classe	Precisão	Sensibilidade	F1
1	0.33	0.29	0.31
2	0.57	0.67	0.62
3	0.00	0.00	0.00
4	0.00	0.00	0.00
5	0.67	1.00	0.80
6	0.78	1.00	0.88
7	1.00	0.25	0.40
8	0.00	0.00	0.00
9	0.00	0.00	0.00
10	0.33	0.81	0.47

Fonte: Elaborada pelo autor

Tabela 4 – Métricas de avaliação time Ri-One para janela de entrada 60

Classe	Precisão	Sensibilidade	F1
1	0.28	0.71	0.40
2	1.00	1.00	1.00
3	0.00	0.00	0.00
4	0.00	0.00	0.00
5	0.89	1.00	0.94
6	1.00	1.00	1.00
7	1.00	1.00	1.00
8	0.00	0.00	0.00
9	1.00	1.00	1.00
10	0.50	0.62	0.56

Fonte: Elaborada pelo autor

Tabela 5 – Métricas de avaliação time Ri-One para janela de entrada 90

Classe	Precisão	Sensibilidade	F1
1	0.22	0.71	0.33
2	1.00	1.00	1.00
3	0.14	0.17	0.15
4	0.17	0.11	0.13
5	0.89	1.00	0.94
6	1.00	1.00	1.00
7	1.00	1.00	1.00
8	0.00	0.00	0.00
9	1.00	0.25	0.40
10	0.70	0.44	0.54

Fonte: Elaborada pelo autor

Tabela 6 – Métricas de avaliação time Ri-One para janela de entrada 120

Classe	Precisão	Sensibilidade	F1
1	0.50	0.86	0.63
2	1.00	1.00	1.00
3	0.80	0.67	0.73
4	0.42	0.56	0.48
5	0.89	1.00	0.94
6	1.00	1.00	1.00
7	1.00	1.00	1.00
8	0.00	0.00	0.00
9	1.00	1.00	1.00
10	0.57	0.50	0.53

Fonte: Elaborada pelo autor

Tabela 7 – Métricas de avaliação time Ri-One para janela de entrada 150

Classe	Precisão	Sensibilidade	F1
1	0.38	0.86	0.52
2	1.00	1.00	1.00
3	0.75	1.00	0.86
4	0.78	0.78	0.78
5	0.89	1.00	0.94
6	1.00	1.00	1.00
7	1.00	1.00	1.00
8	1.00	0.33	0.50
9	1.00	1.00	1.00
10	0.88	0.44	0.58

Fonte: Elaborada pelo autor

Através da tabela de métricas do time Ri-One observa-se que há uma conversão dos valores de precisão, ou seja de todos os valores de classe positivo quantas o modelo acertou, e sensibilidade, ou seja dentre todas de classe positivo como valor esperado, quantas foram classificadas corretas, como neste estudo é mais critico a avaliação dos falsos positivos, focaremos na análise da precisão. Porém ao analisar a classe 1, classe esta que não sofreu o *padding* observou-se uma metrica de precisão não satisfatória, uma vez que métricas não satisfatória foram considerados os valores onde a ultima janela de 150 obtiveram valores inferires a 0,7 , o que pode ser um indício que talvez a rede não tenha aprendido satisfatoriamente e os demais estados onde foram aplicados o *padding* obtiveram métricas satisfatórias, uma vez que métricas satisfatória foram considerados os valores onde a ultima janela de 150 obtiveram valores superiores a 0,7 .

As variáveis utilizadas para treinar o time ITAdroids são: número de estados 322, o número de classes 11, seguida do número máximo de elementos de uma sequência 150. Para o treinamento destas sequências, obteve-se uma acurácia global 40%, para obtenção de

Tabela 8 – Métricas de avaliação time ITAdroids para janela de entrada 30

Classe	Precisão	Sensibilidade	F1
1	0.07	0.14	0.09
2	0.57	0.67	0.62
3	0.00	0.00	0.00
4	1.00	1.00	1.00
5	0.38	1.00	0.56
6	0.57	1.00	0.73
7	1.00	1.00	1.00
8	0.00	0.00	0.00
9	0.00	0.00	0.00
10	0.00	0.00	0.00
11	0.67	0.50	0.57

Fonte: Elaborada pelo autor

Tabela 9 – Métricas de avaliação time ITAdroids para janela de entrada 60

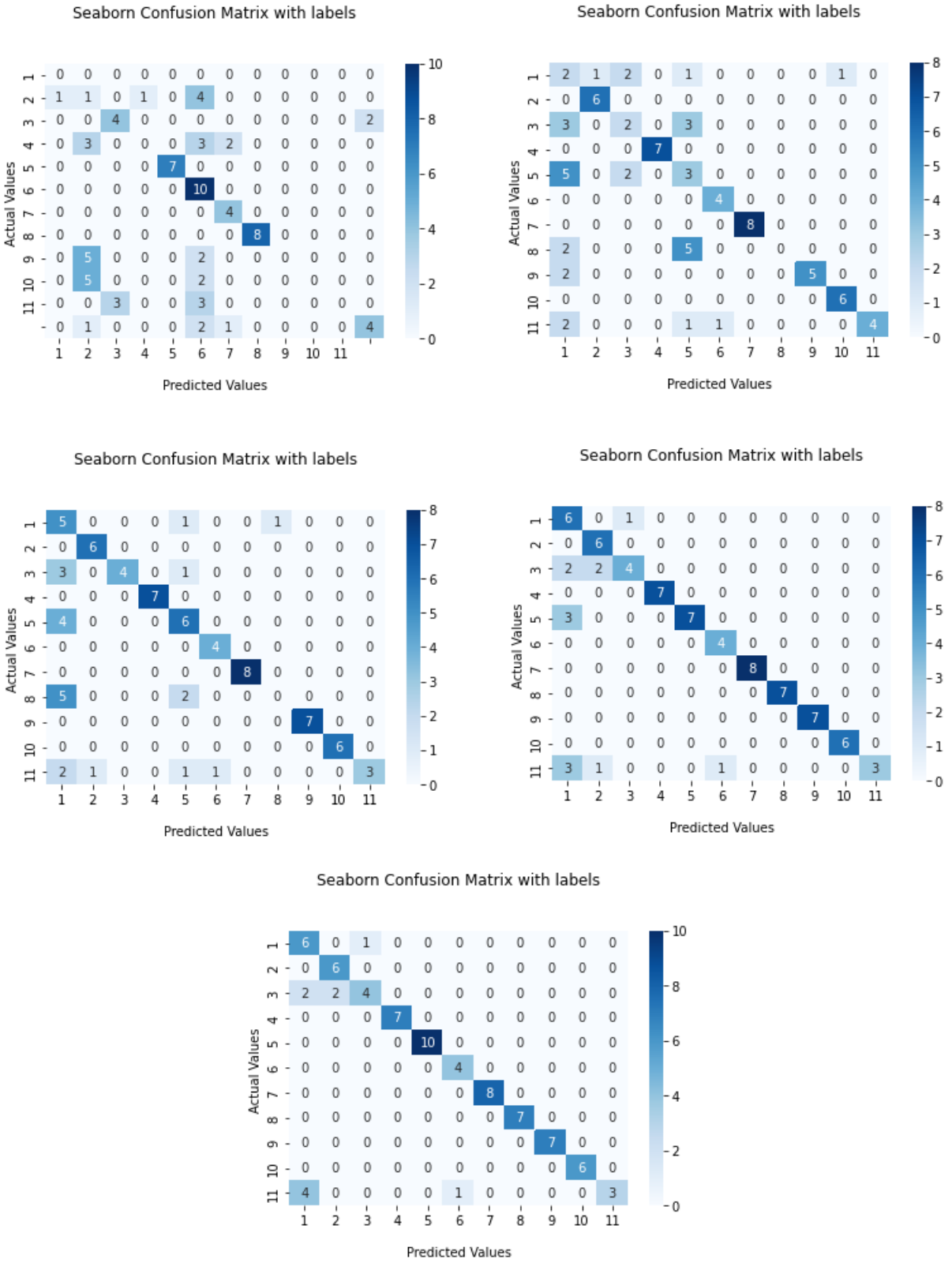
Classe	Precisão	Sensibilidade	F1
1	0.12	0.29	0.17
2	0.86	1.00	0.92
3	0.33	0.25	0.29
4	1.00	1.00	1.00
5	0.23	0.30	0.26
6	0.80	1.00	0.89
7	1.00	1.00	1.00
8	0.00	0.00	0.00
9	1.00	0.71	0.83
10	0.86	1.00	0.92
11	1.00	0.50	0.67

Fonte: Elaborada pelo autor

um melhor resultado foi balanceada as classes, obtendo-se assim uma melhora na acurácia para 87.17%. Para uma análise mais detalhada, utilizando o dataset balanceado, foram geradas as matrizes de confusão das entradas do treinamento considerando a distribuição nas seguintes janelas de entrada 30, 60, 90, 120, 150, conforme a Figura 27 através destas métricas observa-se que com a evolução das janelas a rede converge para as corretas classificações, ou seja nas janelas iniciais observa-se uma maior taxa de erro na classificação dos estados, conforme a janela vai aumentando observa-se a convergência dos valores para a correta classificação. Sendo também gerados os valores de precisão, sensibilidade e F1 de cada uma das janelas, conforme Tabelas 8, 9, 10, 11, 12.

Através da tabela de métricas do time ITAdroids observa-se uma conversão dos valores de precisão e sensibilidade global. Analisando as classes que utilizaram pouco ou nenhum*padding* classes 1 e 2, somente a classe 1 obteve uma convergência não satisfatória ao

Figura 27 – Matriz de confusão do time ITAdroids nas janelas:
(a)30 (b)60 (c)90 (d)120 (e)150



Fonte: Elaborada pelo autor

Tabela 10 – Métricas de avaliação time ITAdroids para janela de entrada 90

Classe	Precisão	Sensibilidade	F1
1	0.26	0.71	0.38
2	0.86	1.00	0.92
3	1.00	0.50	0.67
4	1.00	1.00	1.00
5	0.55	0.60	0.57
6	0.80	1.00	0.89
7	1.00	1.00	1.00
8	0.00	0.00	0.00
9	1.00	1.00	1.00
10	1.00	1.00	1.00
11	1.00	0.38	0.55

Fonte: Elaborada pelo autor

Tabela 11 – Métricas de avaliação time ITAdroids para janela de entrada 120

Classe	Precisão	Sensibilidade	F1
1	0.43	0.86	0.57
2	0.67	1.00	0.80
3	0.80	0.50	0.62
4	1.00	1.00	1.00
5	1.00	0.70	0.82
6	0.80	1.00	0.89
7	1.00	1.00	1.00
8	1.00	1.00	1.00
9	1.00	1.00	1.00
10	1.00	1.00	1.00
11	1.00	0.38	0.55

Fonte: Elaborada pelo autor

Tabela 12 – Métricas de avaliação time ITAdroids para janela de entrada 150

Classe	Precisão	Sensibilidade	F1
1	0.50	0.86	0.63
2	0.75	1.00	0.86
3	0.80	0.50	0.62
4	1.00	1.00	1.00
5	1.00	1.00	1.00
6	0.80	1.00	0.89
7	1.00	1.00	1.00
8	1.00	1.00	1.00
9	1.00	1.00	1.00
10	1.00	1.00	1.00
11	1.00	0.38	0.55

Fonte: Elaborada pelo autor

Tabela 13 – Métricas de avaliação time Helios para janela de entrada 30

Classe	Precisão	Sensibilidade	F1
1	0.40	0.17	0.24
2	0.42	0.50	0.45
3	0.71	0.89	0.79
4	0.94	1.00	0.97
5	0.67	0.63	0.65

Fonte: Elaborada pelo autor

Tabela 14 – Métricas de avaliação time Helios para janela de entrada 60

Classe	Precisão	Sensibilidade	F1
1	0.60	0.50	0.55
2	0.47	0.70	0.56
3	0.85	0.89	0.87
4	0.94	1.00	0.97
5	0.71	0.53	0.61

Fonte: Elaborada pelo autor

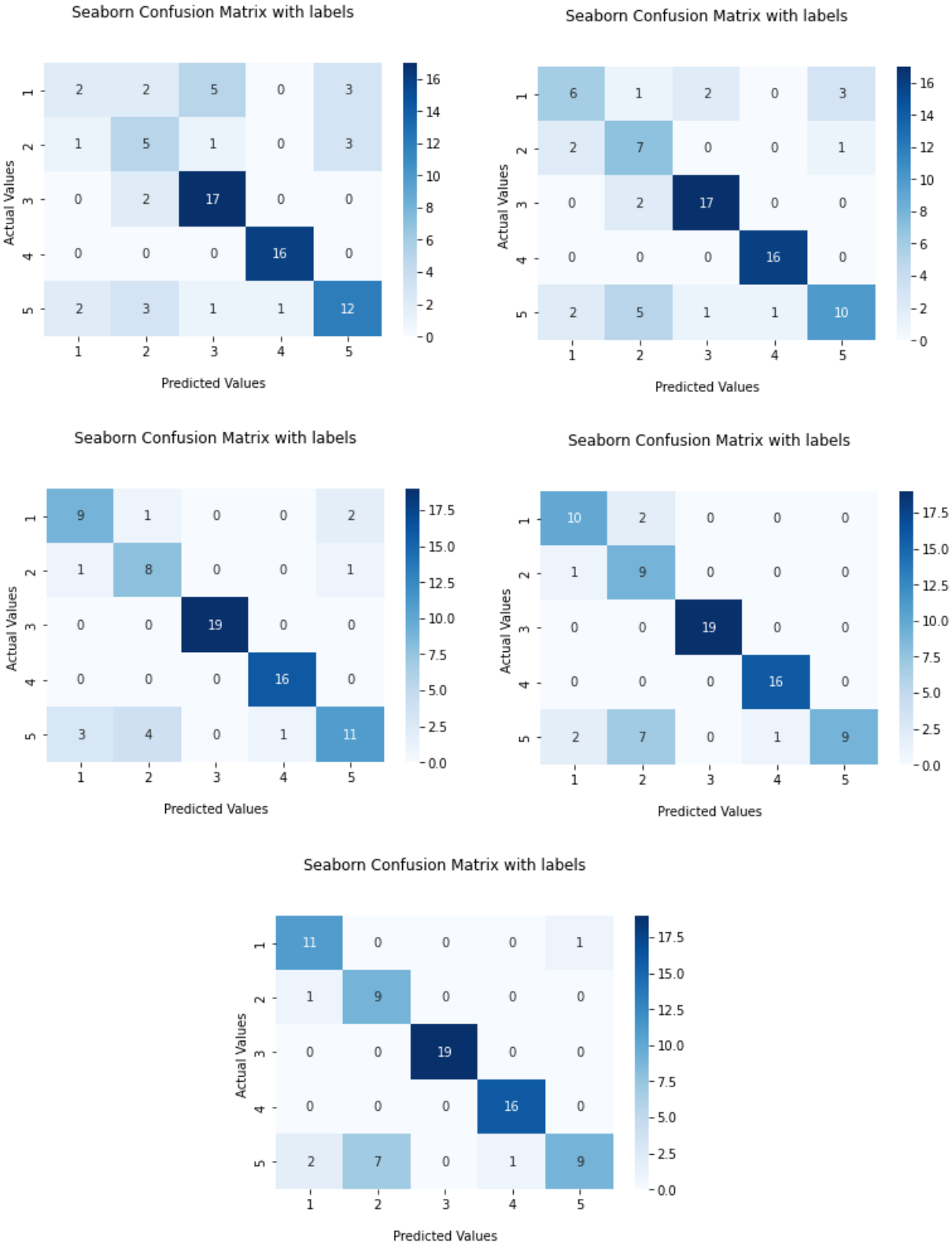
analisar a matriz de métricas de avaliação, as demais classes, classes de 2 a 10, apresentaram uma precisão satisfatória, sendo assim uma indicação que pode caracterizar o aprendizado.

As variáveis utilizadas para treinar o time Helios são: número de estados diferentes na sequências de 275, o número de classes que possui sendo 5, seguida do número máximo de elementos de uma sequência e o tamanho máximo da sequência como 150. Para o treinamento destas sequências, obteve-se uma acurácia global 65,3%, ao aplicar o balanceamento do dataset, apresentando uma melhora na acurácia para 77,63%. Para uma análise mais detalhada, utilizando o dataset balanceado, foram geradas as matrizes de confusão das entradas do treinamento considerando a distribuição nas seguintes janelas de entrada 30, 60, 90, 120, 150, conforme a Figura 28 através destas métricas observa-se que com a evolução das janelas a rede converge para as corretas classificações, ou seja nas janelas iniciais temos uma maior taxa de erro na classificação dos estados, conforme a janela vai aumentando obtém-se a convergência dos valores para a correta classificação. Sendo também gerados os valores de precisão, sensibilidade e F1 de cada uma das janelas, conforme Tabelas 13, 14, 15, 16, 17.

Através da tabela de métricas do time Helios observa-se que também há uma conversão dos valores de precisão e sensibilidade. Analisando as classes que utilizaram pouco ou nenhum*padding* foram as classes 1 e 2, para classe 1 observou-se uma precisão satisfatória, para classe 2 observando-se uma precisão não satisfatória, as demais classes 3 e 4 apresentaram uma precisão e satisfatória, sendo assim uma indicação que pode caracterizar o aprendizado.

Nesta fase de treinamento observou-se que uma quantidade menor de classes geradas

Figura 28 – Matriz de confusão do time Helios nas janelas:
(a)30 (b)60 (c)90 (d)120 (e)150



Fonte: Elaborada pelo autor

Tabela 15 – Métricas de avaliação time Helios para janela de entrada 90

Classe	Precisão	Sensibilidade	F1
1	0.69	0.75	0.72
2	0.62	0.80	0.70
3	1.00	1.00	1.00
4	0.94	1.00	0.97
5	0.79	0.58	0.67

Fonte: Elaborada pelo autor

Tabela 16 – Métricas de avaliação time Helios para janela de entrada 120

Classe	Precisão	Sensibilidade	F1
1	0.77	0.83	0.80
2	0.50	0.90	0.64
3	1.00	1.00	1.00
4	0.94	1.00	0.97
5	1.00	0.47	0.64

Fonte: Elaborada pelo autor

Tabela 17 – Métricas de avaliação time Helios para janela de entrada 150

Classe	Precisão	Sensibilidade	F1
1	0.79	0.92	0.85
2	0.56	0.90	0.69
3	1.00	1.00	1.00
4	0.94	1.00	0.97
5	0.90	0.47	0.62

Fonte: Elaborada pelo autor

na etapa anterior e um balanceamento das mesmas favorecia a acurácia do treinamento. O time que apresentou maior número de jogadas e mais balanceadas foi o time Helios, apresentando um total de 4 classes de interesse e uma delas com elementos não classificados. Uma vez que todos os times possuíam uma matriz de confusão que evoluía para uma matriz diagonal, ou seja convergindo para correta classificação, obtendo também valores de acurácia do modelo satisfatórios e os valores de precisão onde a maioria das classes obtiveram uma precisão satisfatória, consistindo estes em parâmetros os quais evidenciam o aprendizado da rede.

8 CONCLUSÃO

Este trabalho consiste na interação dos caminhos de todos os jogadores mais a bola e modelados como um conjunto de caminhos que possuem interação a fim de atingir um objetivo único, sendo aplicado ao futebol de robôs na categoria de simulação 2D, sendo este o dataset utilizado. Esta proposta foi dividida em 3 objetivos específícos, os quais são: preparação o *dataset* do futebol de robôs simulado, classificação das trajetórias e a previsão da jogada durante a execução do jogo.

Na etapa de classificação das jogadas, a clusterização dos caminhos apresentou resultados que evidenciam uma maior padronização das jogadas do time Helios em comparação aos demais, uma vez que seus clusters gerados concentraram em determinada área do campo e em comparação aos demais times em que possuíam clusters mais dispersos na extensão do campo. Na classificação das jogadas pouco pode ser interpretado sobre os resultados obtidos, uma vez que as sequências encontradas não são de interpretação para o ser humano e constituindo assim como um tipo informação que está em um formato para simplificação do modelo a ser interpretado pela rede neural.

Na etapa de previsão obteve-se resultados que evidenciaram o aprendizado desta rede, através da convergência da matriz de confusão e valores de acurácia e precisão satisfatórios.

Para este trabalho obteve-se o resultado esperado, o qual consiste em apresentar uma ferramenta para predição de jogadas. Sendo alcançada a partir da classificação das jogadas de interesse, lançando-se mão da teoria de máquinas de estados, e realizada a predição, onde dada uma determinada jogada em andamento será obtida a probabilidade de pertencer ou não a uma das jogadas previamente classificadas. Como trabalhos futuros para uma melhor validação destes resultados serão realizadas as análise com uma maior quantidade de dados, bem como realizar a implementação deste algorimo na estratégia online.

REFERÊNCIAS

ABELLA, A.; WRIGHT, J.; GORIN, A. L. Dialog trajectory analysis. In: **2004 IEEE International Conference on Acoustics, Speech, and Signal Processing**. [S.l.: s.n.], 2004. v. 1, p. I–441. ISSN 1520-6149.

AGARWAL, P. K. et al. Computing the discrete frÉchet distance in subquadratic time. In: **Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms**. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2013. p. 156–167. ISBN 978-1-611972-51-1.

ALAHI, A. et al. Social lstm: Human trajectory prediction in crowded spaces. In: **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2016. p. 961–971.

AMMOUN, S.; NASHASHIBI, F. Real time trajectory prediction for collision risk estimation between vehicles. In: **2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing**. [S.l.: s.n.], 2009. p. 417–422.

ARONOV, B. et al. Fréchet Distance for Curves, Revisited. In: AZAR, Y.; ERLEBACH, T. (Ed.). **Algorithms – ESA 2006**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 52–63. ISBN 978-3-540-38876-0.

ASALI, E. et al. Using machine learning approaches to detect opponent formation. In: **2016 Artificial Intelligence and Robotics (IRANOPEN)**. [S.l.: s.n.], 2016. p. 140–144.

BARBOSA, H. J. **Identification Plays In Robot Soccer**. 2022. <https://github.com/heloisajunqueira/IdentificationPlaysInRobotSoccer>(acessado em Juhho, 2022).

BERGAMASCHI, R. A.; CAMPOSANO, R.; MICHAEL, P. Data-path synthesis using path analysis. In: **28th ACM/IEEE Design Automation Conference**. [S.l.: s.n.], 1991. p. 591–596.

BIAN, J. et al. A survey on trajectory clustering analysis. 2018. Disponível em: <http://arxiv.org/abs/1802.06971>.

BISHOP, C. M. **Pattern Recognition and Machine Learning (Information Science and Statistics)**. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0387310738.

BUCHIN, K. et al. Clustering Trajectories for Map Construction. p. 1–10, 2018.

_____. Detecting Commuting Patterns by Clustering Subtrajectories. In: HONG, S.-H.; NAGAMOCHI, H.; FUKUNAGA, T. (Ed.). **Algorithms and Computation**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 644–655. ISBN 978-3-540-92182-0.

_____. Computing the fréchet distance with a retractable leash. In: BODLAENDER, H. L.; ITALIANO, G. F. (Ed.). **Algorithms – ESA 2013**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 241–252. ISBN 978-3-642-40450-4.

CAI, L. et al. GPS Trajectory Clustering and Visualization Analysis. **Annals of Data Science**, v. 5, n. 1, p. 29–42, mar 2018. ISSN 2198-5812. Disponível em: <https://doi.org/10.1007/s40745-017-0131-2>.

CASSIANO, K. M. **Análise de Séries Temporais Usando Análise Espectral Singular (SSA) e Clusterização de Suas Componentes Baseada em Densidade**. 2014. Tese (Doutorado) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, 2014.

CHOI, I.; SONG, H.; YOO, J. Deep Learning Based Pedestrian Trajectory Prediction Considering Location Relationship between Pedestrians. **1st International Conference on Artificial Intelligence in Information and Communication, ICAIIC 2019**, IEEE, p. 449–451, 2019.

CHUNG, J. et al. Gated feedback recurrent neural networks. **32nd International Conference on Machine Learning, ICML 2015**, v. 3, p. 2067–2075, 2015.

DAS, M.; LERNER, S.; SEIGLE, M. Esp: Path-sensitive program verification in polynomial time. **SIGPLAN Not.**, ACM, New York, NY, USA, v. 37, n. 5, p. 57–68, maio 2002. ISSN 0362-1340. Disponível em: <http://doi.acm.org/10.1145/543552.512538>.

ENDSLEY, E. W.; LUCAS, M. R.; TILBURY, D. M. Software tools for verification of modular fsm based logic control for use in reconfigurable machining systems. In: . [S.l.: s.n.], 2000.

FLASH, T. The control of hand equilibrium trajectories in multi-joint arm movements. **Biol. Cybern.**, Springer-Verlag, Berlin, Heidelberg, v. 57, n. 4–5, p. 257–274, nov 1987. ISSN 0340-1200. Disponível em: <https://doi.org/10.1007/BF00338819>.

FU, Z.; HU, W.; TAN, T. Similarity based vehicle trajectory clustering and anomaly detection. In: **Proceedings - International Conference on Image Processing, ICIP**. [S.l.: s.n.], 2005. ISBN 0780391349. ISSN 15224880.

GIRGIS, H. Z.; JAMES, B. T.; LUCZAK, B. B. Identity: rapid alignment-free prediction of sequence alignment identity scores using self-supervised general linear models. **NAR Genomics and Bioinformatics**, v. 3, n. 1, 02 2021. ISSN 2631-9268. Lqab001. Disponível em: <https://doi.org/10.1093/nargab/lqab001>.

_____. **Identity MeShClust**. 2022. <https://github.com/BioinformaticsToolsmith/Identity>(acessado em Junho, 2022).

GUDMUNDSSON, J.; WOLLE, T. Football analysis using spatio-temporal tools. **Computers, Environment and Urban Systems**, Elsevier Ltd, v. 47, p. 16–27, 2014. ISSN 01989715. Disponível em: <http://dx.doi.org/10.1016/j.compenvurbsys.2013.09.004>.

HAN, J.; KAMBER, M.; PEI, J. **Data Mining: Concepts and Techniques**. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN 0123814790, 9780123814791.

HONG, P.; TURK, M.; HUANG, T. S. Gesture modeling and recognition using finite state machines. In: **Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)**. [S.l.: s.n.], 2000. p. 410–415.

HUANG, C.-C. **DISCRETE EVENT SYSTEM MODELING USING SYSML AND MODEL TRANSFORMATION**. 2011. Dissertação (Mestrado) — Georgia Institute of Technology, Atlanta, GA, USA, 2011.

JACKSON, M. R.; ZHAO, Y. J.; SLATTERY, R. A. Sensitivity of trajectory prediction in air traffic management. **Journal of Guidance, Control, and Dynamics**, v. 22, n. 2, p. 219–228, 1999.

JO, K. et al. Development of autonomous car — part ii: A case study on the implementation of an autonomous driving system based on distributed architecture. **IEEE Transactions on Industrial Electronics**, v. 62, n. 8, p. 5119–5132, Aug 2015. ISSN 0278-0046.

KEMPOWSKY, T.; SUBIAS, A.; AGUILAR-MARTIN, J. Process situation assessment: From a fuzzy partition to a finite state machine. **Engineering Applications of Artificial Intelligence**, v. 19, n. 5, p. 461–477, 2006. ISSN 0952-1976. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0952197606000194>.

KERAS. 2022. <https://keras.io/>(acessado em Junho, 2022).

KERFS, J. **Models for Pedestrian Trajectory Prediction and Navigation in Dynamic Environments**. 2017. Dissertação (Mestrado) — Faculty of California Polytechnic State University, San Luis Obispo, CA, USA, 2017.

KUMMENEJE, J. RoboCup as a Means to Research , Education , and Dissemination. **Technology**, 2003.

KURT, A.; ÖZGÜNER, Ü. Hierarchical finite state machines for autonomous mobile systems. **Control Engineering Practice**, v. 21, n. 2, p. 184–194, 2013. ISSN 0967-0661. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0967066112002031>.

LANG, R. G.; SILVA, I. N.; ROMERO, R. A. Development of distributed control architecture for multi-robot systems. In: **2014 Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol**. [S.l.: s.n.], 2014. p. 163–168.

LEE, D.; YANNAKAKIS, M. Principles and methods of testing finite state machines - A survey. **Proceedings of the IEEE**, v. 84, n. 9, p. 1090–1123, 1996. ISSN 1098-6596.

LEE, J.-G.; HAN, J.; WHANG, K.-Y. Trajectory clustering: A partition-and-group framework. **Proceedings of the ACM SIGMOD International Conference on Management of Data**, 01 2007.

LI, Z. T.; HE, Z. C.; ZHAO, J. M. Short-term traffic flow prediction with nearest trajectory segments. **2010 2nd International Conference on Computational Intelligence and Natural Computing, CINC 2010**, IEEE, v. 2, p. 312–315, 2010.

LIPTON, Z. C.; BERKOWITZ, J.; ELKAN, C. A Critical Review of Recurrent Neural Networks for Sequence Learning. p. 1–38, 2015. Disponível em: <http://arxiv.org/abs/1506.00019>.

MACKWORTH, A. K. **On Seeing Robots**. Vancouver, BC, Canada, Canada, 1993.

MAO, Y. et al. An adaptive trajectory clustering method based on grid and density in mobile pattern analysis. **Sensors (Switzerland)**, 2017. ISSN 14248220.

MARSLAND, S. **Machine Learning: An Algorithmic Perspective, Second Edition**. 2nd. ed. [S.l.]: Chapman & Hall/CRC, 2014. ISBN 1466583282, 9781466583283.

MELLO, R. F.; PONTI, M. A. **Machine Learning: A Practical Approach on the Statistical Learning Theory**. [S.l.]: Springer International Publishing, 2018. v. 1. ISBN 9783319949888.

MEULEMANS, W. 2019. <http://www.win.tue.nl/~wmeulema/implementations.html>(acessado em Julho, 2019).

MICHAEL, O. et al. Analysing soccer games with clustering and conceptors. In: AKIYAMA, H. et al. (Ed.). **RoboCup 2017: Robot World Cup XXI**. Cham: Springer International Publishing, 2018. p. 120–131. ISBN 978-3-030-00308-1.

MURCA, M. C. R. et al. Trajectory clustering and classification for characterization of air traffic flows. 2016. Disponível em: <https://arc.aiaa.org/doi/abs/10.2514/6.2016-3760>.

OSTANIN, S. Self-checking synchronous fsm network design for path delay faults. In: **2017 IEEE East-West Design Test Symposium (EWDTS)**. [S.l.: s.n.], 2017. p. 1–4. ISSN 2472-761X.

PERERA, L. P.; OLIVEIRA, P.; SOARES, C. G. Maritime traffic monitoring based on vessel detection, tracking, state estimation, and trajectory prediction. **IEEE Transactions on Intelligent Transportation Systems**, v. 13, n. 3, p. 1188–1200, Sep. 2012.

PREVOST, C. G.; DESBIENS, A.; GAGNON, E. Extended kalman filter for state estimation and trajectory prediction of a moving object detected by an unmanned aerial vehicle. In: **2007 American Control Conference**. [S.l.: s.n.], 2007. p. 1805–1810. ISSN 0743-1619.

QIAO, S. et al. A self-adaptive parameter selection trajectory prediction approach via hidden markov models. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 16, n. 1, p. 284–296, 2015. ISSN 15249050.

REIS, L. P. **Coordenação em Sistemas Multi-Agente: Aplicações na Gestão Universitária e Futebol Robótico**. 2003. Tese (Doutorado) — Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2003.

REIS, L. P.; LAU, N.; OLIVEIRA, E. C. Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents. In: **Balancing Reactivity and Social Deliberation in Multi-Agent Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. p. 175–197. ISBN 978-3-540-44568-5.

RIEDMILLER, M. et al. Karlsruhe Brainstormers - A Reinforcement Learning approach to robotic soccer. In: STONE, P.; BALCH, T.; KRAETZSCHMAR, G. (Ed.). **RoboCup 2000: Robot Soccer World Cup IV**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. p. 367–372. ISBN 978-3-540-45324-6.

ROBOCUP. **RoboCup Federation official website**. 2018. <https://www.robocup.org/>(acessado em Julho, 2018).

_____. **The RoboCup Soccer Simulator Users Manual**. 2019. <https://rcsoccersim.github.io/manual/>(acessado em Julho, 2019).

_____. **RoboCup.info Archive**. 2019. <https://archive.robocup.info/Soccer/Simulation/2D/logs/RoboCup/>(acessado em Julho, 2019).

SCHUBERT, E. et al. Dbscan revisited, revisited: Why and how you should (still) use dbscan. **ACM Trans. Database Syst.**, Association for Computing Machinery, New York, NY, USA, v. 42, n. 3, jul 2017. ISSN 0362-5915. Disponível em: <https://doi.org/10.1145/3068335>.

ULLMAN, D. S. et al. Trajectory prediction using hf radar surface currents: Monte carlo simulations of prediction uncertainties. **Journal of Geophysical Research: Oceans**, v. 111, n. C12, 2006.

VERMA, R.; DEV, A. Vision based hand gesture recognition using finite state machines and fuzzy logic. **2009 International Conference on Ultra Modern Telecommunications and Workshops**, IEEE, p. 1–6, 2009.

WAGNER, F. et al. **Modeling Software with Finite State Machines**. Boston, MA, USA: Auerbach Publications, 2006. ISBN 0849380863.

WIEST, J. et al. Probabilistic trajectory prediction with gaussian mixture models. In: **2012 IEEE Intelligent Vehicles Symposium**. [S.l.: s.n.], 2012. p. 141–146. ISSN 1931-0587.

WILLMOTT, C. J.; MATSUURA, K. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. **Climate Research**, Inter-Research Science Center, v. 30, n. 1, p. 79–82, 2005. ISSN 0936577X, 16161572. Disponível em: <http://www.jstor.org/stable/24869236>.

WOOLDRIDGE, M. **An Introduction to MultiAgent Systems**. 2nd. ed. [S.l.]: Wiley Publishing, 2009. ISBN 0470519460, 9780470519462.

XU, Y.; PIAO, Z.; GAO, S. Encoding Crowd Interaction with Deep Neural Network for Pedestrian Trajectory Prediction. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, IEEE, p. 5275–5284, 2018. ISSN 10636919.

YARN, Z. **RoboCup 2D Log Mining**. 2018. <https://github.com/zenoyarn/2DLogMining>(acessado em Julho, 2019).

YASUTOMI, F.; YAMADA, M.; TSUKAMOTO, K. Cleaning robot control. In: **Proceedings. 1988 IEEE International Conference on Robotics and Automation**. [S.l.: s.n.], 1988. p. 1839–1841 vol.3.

YEPES, J. L.; HWANG, I.; ROTEA, M. New algorithms for aircraft intent inference and trajectory prediction. **Journal of Guidance, Control, and Dynamics**, v. 30, n. 2, p. 370–382, 2007.

YU, Y. et al. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. **Neural Computation**, v. 31, n. 7, p. 1235–1270, 07 2019. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/neco_a_01199>.

YUAN, G. et al. A review of moving object trajectory clustering algorithms. **Artificial Intelligence Review**, Springer Netherlands, v. 47, n. 1, p. 123–144, 2017. ISSN 15737462.

ZHANG, W. et al. Trajectory Prediction with Recurrent Neural Networks for Predictive Resource Allocation. **International Conference on Signal Processing Proceedings, ICSP**, IEEE, v. 2018-Augus, p. 634–639, 2019.

ZHANG, Z.; YANG, R.; FANG, Y. LSTM Network Based on on Antlion Optimization and its Application in Flight Trajectory Prediction. **Proceedings of 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference, IMCEC 2018**, IEEE, n. Imcec, p. 1658–1662, 2018.