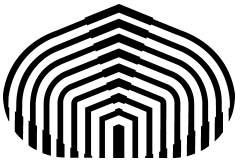


Universidad Simón Bolívar
Decanato de Estudios Profesionales
Coordinación de Ingeniería de la Computación

Coach para equipos de la liga simulada 2D de RoboCup

Por:
María Victoria Jorge Mauriello
Jorge Enrique Marcano Dutkowski

PROYECTO DE GRADO
Presentado ante la Ilustre Universidad Simón Bolívar
como requisito parcial para optar al título de
Ingeniero de Computación

Sartenejas, marzo de 2017

Resumen

Este documento describe el desarrollo de un coach para un equipo de futbol de la liga de simulacion 2D de RoboCup. Se elaboraron dos funcionalidades relacionadas al coach que buscan mejorar el desempeño del equipo: una consiste en la detección de acciones básicas realizadas durante el partido y el uso de árboles de decisión para predecir si serán exitosas; la otra le permite identificar la formación oponente y desarrollar estrategias para contrarrestarla. Los resultados obtenidos son producto de una prueba de tipo t de Student con 250 partidos de control y 250 experimentales y con diferencia de goles para cada equipo como métrica de desempeño. Usando los árboles de decisión, el desempeño del equipo empeora a pesar de que la clasificacion de los árboles es correcta en un 60-70 % de los casos dependiendo del modelo, pasando de una media de diferencia de goles de -4,816 a una de -6,94, siendo este un deterioro estadísticamente signicativo en el desempeño del equipo. Por otro lado, se intentó hacer uso de redes neuronales para la predicción de acciones, pero los resultados de estas tampoco fueron satisfactorios. Se puede concluir que el uso de estos modelos no debe limitarse a la predicción correcta de acciones. Es necesario plantear alternativas para los casos en los que las predicciones son negativas. Al utilizar la detección de formaciones y planteamiento de estrategias mejora considerablemente el desempeño. Se pasa de una media de diferencia de goles de -4,816 a una de -2,372, ocasionando una reducción estadísticamente signicativa en la diferencia de goles y por lo tanto una mejora para el equipo. La combinación de ambas funcionalidades tambien mejora su desempeño, pero en menor medida que usando únicamente la detección de formaciones. En este caso se pasa de una media de -4,816 a una de -3,94. Esto se considera una reducción estadísticamente signicativa en las medias y una mejora en el desempeño.

Palabras clave: inteligencia artificial, sistemas multi-agente, robocup, fútbol.

Agradecimientos

Agradecemos a nuestros padres y familiares, Alejandra Dutkowski, Jorge Marcano, Amalia Mauriello, Carlos Jorge y María Emilia Jorge, por ser pilares fundamentales durante nuestra vida, pero en especial durante este año, estando siempre cuando los necesitamos.

A nuestra tutora, la profesora Carolina Chang, por aceptar acompañarnos en este camino y brindarnos el apoyo, los conocimientos y la, a veces, tan necesaria calma hasta el último momento de este proyecto.

A los profesores Ángela Di Serio, Pedro Ovalles y Minaya Villasana, por brindarnos su apoyo y orientación en el área de estadística.

A nuestros amigos, Mónica Figuera, Patricia Reinoso y Richard Lares, por acompañarnos desde el inicio de la carrera, por los días de estudio, las noches de desvelo terminando proyectos y por todos los buenos momentos vividos en estos años. Sin ustedes este tiempo de nuestra carrera no hubiese sido el mismo.

A Maryam Karimi, Greg Kuhlmann, Patrick MacAlpine, Mikhail Prokopenko, Patrick Riley y Peter Stone, todos participantes de la comunidad RoboCup, que respondieron de forma desinteresada a nuestras dudas, proporcionándonos consejos útiles para la comprensión de la liga de simulación.

A Manuel Gomes, Rafael Rojas y Genessis Sánchez, por estar dispuestos siempre a escuchar nuestras ideas futbolísticas.

A Frank Vicente, entrenador de los equipos de fútbol sala de la USB, y a las compañeras del equipo femenino, por ser fuente de inspiración para nuestro coach y jugadores.

Al Laboratorio Docente de Aulas Computarizadas (MAC), al Grupo de Inteligencia Artificial y a sus integrantes, por permitirnos utilizar sus instalaciones. Pero aún más importante, por ser nuestra segunda familia. Nos permitieron crecer más allá de lo que ofrecen las materias de la carrera y se preocuparon en todo momento por nosotros y nuestra tesis.

Por último, pero no menos importante, queremos agradecer a nuestra *Alma Mater*, la Universidad Simón Bolívar, por todo lo que nos ha brindado y por enseñarnos que aún en las adversidades se pueden lograr grandes cosas.

Índice general

Índice de figuras

Índice de tablas

Introducción

Los deportes de equipo pueden ser utilizados en muchas áreas de investigación de Inteligencia Artificial como un problema estándar para evaluar teorías, algoritmos y arquitecturas de modelos, puesto que poseen ambientes dinámicos, transcurren en tiempo real; la información del ambiente y los jugadores se obtiene de manera incompleta y el sistema de control es distribuido, ya que los agentes que conforman los equipos deben ser inteligentes y autónomos.

El fútbol es uno de estos deportes donde se tienen dos equipos de 11 jugadores cada uno que buscan realizar una misma tarea: anotar más goles que el oponente en un determinado período de tiempo.

Visto desde la perspectiva de la Inteligencia Artificial, el fútbol puede ser modelado como un problema de sistemas multi-agentes heterogéneos cuya tarea principal a resolver es la de anotar goles. Esto lleva a diferentes interrogantes que deben ser respondidas para construir un buen equipo de fútbol: cómo reconocer las fortalezas y debilidades de ambos equipos para explotar unas y contrarrestar otras, cómo modelar el comportamiento de los agentes involucrados y cómo mejorar su desempeño usando experiencia obtenida de la interacción con el ambiente.

RoboCup (RoboCup, 2017) es una iniciativa internacional cuya misión original era la de desarrollar un equipo de robots capaz de ganar un partido de fútbol contra el equipo humano de fútbol que resulte campeón del mundo en el año 2050. Dentro de las categorías actuales se encuentra la Liga de Simulación 2D de *Robocup Soccer*. En ella hay dos equipos de agentes autónomos que se enfrentan en un partido de fútbol dentro de un ambiente de simulación en 2D.

Este proyecto se enfocará en el desarrollo de un *coach* (director técnico) para un

equipo de la liga simulada 2D de Robocup, buscando, por medio de métodos basados en Inteligencia Artificial y estadística, mejorar el desempeño de dicho equipo en partidos en tiempo real contra distintos equipos participantes de competencias pasadas de Robocup 2D.

Antecedentes

Actualmente existen diversas técnicas para tutelar y sugerir acciones a agentes inteligentes. En el trabajo presentado por (Bieger et al., 2014) aparecen varias técnicas que pueden ayudar a mejorar el aprendizaje en sistemas inteligentes: heurísticas para recompensas, descomposición de tareas, simplificación de la tarea, selección de situaciones, teleoperación, enseñanza por demostración, *coaching*, explicaciones y cooperación.

La técnica de *coaching* involucra a un agente que permite hacer sugerencias a otros agentes de Inteligencia Artificial para mejorar su desempeño en el ámbito que se encuentren. El uso de agentes de este tipo para mejorar el desenvolvimiento de otros ha sido investigado en distintas áreas de la Inteligencia Artificial, como videojuegos en el caso de (Taylor et al., 2014) quienes utilizan un *coach* para mejorar el desempeño de agentes que juegan *Starcraft* y *Pacman*, o para mejorar un proceso de aprendizaje por imitación como es demostrado por (He et al., 2012).

Entre 2001 y 2006 se realizó una competencia dentro de RoboCup enfocada únicamente en la investigación para modelar oponentes en un ambiente dinámico con equipos multi-agente, llamada *RoboCup Coach Simulation Competition*. En lugar de crear un equipo entero solo era necesario implementar un agente que haría las funciones de un *coach* y sería evaluado con base en cómo identificaba las fortalezas y debilidades de los oponentes.

Algunas de las investigaciones realizadas para esa competencia incluyen el uso de reconocimiento de patrones para detectar las estrategias utilizadas por el equipo oponente (Kuhlmann et al., 2006) y el uso de Sistemas Expertos para la toma de decisiones durante un partido (Fathzadeh et al., 2006).

Dentro de la Liga de Simulación 2D de fútbol de RoboCup se poseen trabajos que involucran tanto a jugadores como al *coach*. Uno de ellos trata sobre la detección y clasificación de acciones durante los partidos (Karimi y Ahmazadeh, 2014), que permite mejorar la toma de decisiones de los jugadores dependiendo del rival al que se enfrenten. Un enfoque parecido es el utilizado por el equipo CAOS (Iglesias et al., 2009); en él se extraen eventos del partido usando la posesión de la pelota en cada ciclo y se estudia la dependencia entre ellos.

En esta categoría también se han realizado avances en el área de modelado de los oponentes a partir del reconocimiento de las formaciones de los equipos. Una de las variantes para la detección de los esquemas tácticos de los equipos es el planteado en (Habibi et al., 2002). Estos autores registran la posición relativa de cada jugador respecto al campo durante cada ciclo del partido y luego usan los datos extraídos para obtener la formación del oponente. Otro método para detectar las formaciones de equipos es implementado en (Asali et al., 2016), donde se emplea un método de Inteligencia Artificial para crear un modelo que permita detectar la formación del equipo oponente.

Planteamiento del problema

Uno de los aspectos más importantes de la implementación de equipos de la Liga de Simulación 2D de RoboCup es el modelaje de los sistemas multi-agentes. Delegar esta tarea sobre los propios jugadores dificulta el llegar a un consenso sobre el comportamiento del oponente, puesto que es necesario unificar el punto de vista de 11 jugadores que reciben información constante del campo de juego, sin estar exentos de obtenerla con ruidos propios de las limitaciones físicas que poseen.

Esta investigación se centrará en esto desde la perspectiva del *coach*. Se intentará responder a las preguntas de cómo reconocer las características de un equipo, cómo modelar el comportamiento del equipo que maneja el *coach* a partir de observaciones de otros y cómo contrarrestar el comportamiento del oponente, esperando que estas respuestas permitan mejorar el desenvolvimiento de sus jugadores.

Este proyecto busca diseñar e implementar un *coach* para un equipo básico ya existente de Robocup Soccer 2D, conocido como *Agent2D Base*, creado por el equipo japonés Helios (Behnke et al., 2014). El aporte que realiza el *coach* del equipo base no representa un cambio significativo dentro del desempeño de sus jugadores. Sus labores se limitan a acciones básicas como la determinación del tipo de jugadores a los que se está enfrentando y la sustitución de agentes dentro de su equipo.

El equipo *Agent2D Base* tampoco toma ventaja de los datos que provee el servidor de RoboCup sobre partidos anteriores, aunque estos representen una fuente importante de información tanto para modelar el comportamiento oponente como para mejorar la toma de decisiones de los jugadores. Con el fin de aprovechar esta información se elaborarán modelos basados en Inteligencia Artificial que permitan aprender de la experiencia. En principio se utilizarán árboles de decisión. En modo de comparación de resultados se hará uso también de redes neuronales.

Otra destreza que no se encuentra implementada dentro del *coach* base, y que sus jugadores tampoco poseen, es el análisis del esquema táctico oponente. Más allá de poder modelar las reacciones de los jugadores oponentes ante determinadas situaciones, conocer la formación que está siendo empleada en diferentes momentos del partido puede otorgarle al equipo información crucial para plantear una estrategia de juego.

El *coach* implementado tendrá dos nuevas funcionalidades principales: detección de acciones durante los partidos que permitirán a los jugadores realizar predicciones antes de realizar movimientos, y el reconocimiento de formaciones de los equipos oponentes para mejorar el sistema táctico.

En síntesis, el problema que se plantea está relacionado con la labor del *coach* a la hora de guiar y aconsejar a sus jugadores durante los partidos.

Justificación

Cuando se habla de fútbol, las miradas suelen dirigirse a los jugadores de los equipos y al desempeño que estos realizan dentro de la cancha. Sin embargo, es en

el *coach* en quien recaen algunas de las decisiones más importantes que marcarán el desenvolvimiento final de sus jugadores. La clave para un buen sistema táctico que aproveche al máximo las características del equipo está en conocer no solo las fortalezas y debilidades del equipo que se maneja sino también las del rival de turno.

Esta tarea recae en el *coach* porque es la persona que tiene la visión más amplia del juego y de lo que ocurre en todo momento dentro del campo. En el ambiente de la Liga de Simulación de RoboCup ocurre lo mismo. El *coach* es el único agente del equipo capaz de recibir información completa de los jugadores y del entorno, libre de todo tipo de ruido. Por esto suele utilizarse este agente como el cerebro para el planteamiento de estrategias y modelado del comportamiento del oponente. Al tener un agente encargado únicamente de monitorear a ambos equipos y extraer información relevante de sus comportamientos, podría potenciarse el desempeño del equipo al que pertenece.

El presente trabajo busca potenciar la figura del *coach* aprovechando las facilidades que provee este agente para construir un equipo de fútbol más completo. Debe entenderse como un aporte más en la ampliación del conocimiento dentro del área de investigación sobre agentes que tutelan el comportamiento de otros.

Sin embargo, esta investigación no es útil únicamente dentro de la categoría de simulación 2D. Trabajos como este han sido utilizados para mejorar el desempeño de los robots en otras categorías de RoboCup, como por ejemplo en equipos de la categoría *Middle-Size* (Lau et al., 2010). También podría aplicarse a otros problemas que involucren sistemas multi-agentes heterogéneos que se desenvuelvan en ambientes de confrontación. En este sentido (Taylor et al., 2014) describe un agente inteligente que es aconsejado por otro, que hace el papel de entrenador, para mejorar su desempeño en un caso específico del juego *Starcraft*. El agente aconsejado debe enfrentarse a otro que es controlado por la Inteligencia Artificial que tiene por defecto el juego.

Objetivos

En esta sección se darán a conocer los objetivos del proyecto que se llevará a cabo. Se comienza con el objetivo general para dar una idea del alcance del trabajo y posteriormente se sigue el conjunto de objetivos específicos que se desean alcanzar. Los objetivos específicos planteados en esta sección abarcan los detalles asociados a las funcionalidades que se implementarán en la construcción del *coach*.

Objetivo general

El objetivo general de este trabajo de investigación es diseñar e implementar un *coach* (director técnico) para el equipo base *Agent2D Base* de la liga simulada 2D de RoboCup, con la finalidad de mejorar el desempeño de los jugadores de su equipo durante partidos de fútbol.

Objetivos específicos

- Implementar técnicas que permitan extraer acciones básicas (dribles, pases y tiros al arco) de *logs* de partidos anteriores de los oponentes y del equipo del *coach*.
- Generar modelos basados en técnicas de Inteligencia Artificial que permitan predecir si una acción que será ejecutada en determinado momento del partido será exitosa o no, todo ello a partir de las acciones extraídas durante partidos anteriores.
- Desarrollar técnicas para detectar la formación promedio de los equipos oponentes, tanto en tiempo real como en *logs* de partidos pasados.
- Diseñar estrategias para intentar contrarrestar la formación oponente.
- Determinar si existe diferencia entre el desempeño del equipo que se sirve del *coach* y del equipo que no lo posee.

Estos objetivos se alcanzarán en cuatro capítulos. En el primer capítulo se encuentran las bases teóricas que dieron pie a la realización del proyecto. Esta

parte conceptual presenta un acercamiento a las técnicas de Inteligencia Artificial generalmente utilizadas, al mundo de RoboCup y a las plataformas y herramientas que sirvieron de base para el equipo y *coach* implementado.

En el capítulo 2 se presenta el diseño y la implementación del *coach* explicada en detalle. El capítulo 3 contiene los experimentos y resultados obtenidos con la evaluación de los modelos propuestos, y las pruebas estadísticas para determinar si existe una mejora en el equipo al añadirle las dos nuevas funcionalidades del *coach*. Finalmente, en el capítulo 4 se presentan las conclusiones a las que se llegó con la realización de este trabajo, junto a las recomendaciones para futuros proyectos en esta área.

Capítulo 1

Marco Teórico

Este capítulo contiene las bases teóricas fundamentales para el desarrollo de la presente investigación. En la sección 1.1 se presentan las definiciones de términos relacionados con el área de Inteligencia Artificial y los algoritmos del área que serán utilizados. La sección 1.2 abarca lo relacionado a Sistemas Multiagentes. Luego, en la sección 1.3 se presenta la competencia *RoboCup*, esencial para el desarrollo de este proyecto, además de la plataforma de simulación presente en la sección 1.4. Finalmente, se presentarán las características del equipo base que será empleado para interactuar con el *coach* que se busca construir.

1.1. Inteligencia Artificial

El término 'Inteligencia Artificial' ha variado su definición a lo largo de los años. En (Dean et al., 1995) definen la Inteligencia Artificial como el diseño y estudio de programas de computadora que se comportan de forma inteligente. Por otro lado, en (Russell y Norvig, 2003) dividen las distintas definiciones existentes de este témino en cuatro categorías: sistemas que piensan como humanos, sistemas que actúan como humanos, sistemas que piensan racionalmente y sistemas que actúan racionalmente.

1.1.1. Aprendizaje de Máquinas

El Aprendizaje de Máquinas es un área de la Inteligencia Artificial enfocada en estudiar y crear programas que aprendan y mejoren automáticamente con algún tipo de experiencia. "Se dice que un programa aprende de una experiencia E con respecto a cierta tarea T y métrica de desempeño P, si su desempeño en T, evaluado con P, aumenta con la experiencia E"(Mitchell, 1997).

1.1.2. Árboles de Decisión

El aprendizaje con árboles de decisión es un método del aprendizaje de máquinas que permite aproximar funciones objetivo con valores discretos. La función que aprendió la máquina es representada por un árbol de decisión, que no es más que un árbol de búsqueda que retorna una decisión final.

La representación del árbol de decisión viene dada por las siguientes condiciones (Mitchell, 1997):

- Cada nodo del árbol corresponde con una prueba de uno de los atributos.
- Los arcos entre un nodo padre y sus nodos hijos representan los valores que puede tomar el atributo presente en el padre.
- Cada hoja del árbol constituye el valor booleano retornado en caso de que la búsqueda haya llegado hasta ese punto.

Esta técnica de aprendizaje ha sido utilizada para resolver problemas de diferente índole. Sin embargo, en (Mitchell, 1997) proponen una serie de condiciones que debe cumplir un problema si se desea obtener el máximo provecho por este método de aprendizaje:

- Las instancias tienen que estar representadas por un par atributo-valor. Además, cada atributo debe estar representado por un número fijo de valores. Sin embargo, existen modificaciones en los algoritmos de árboles de decisión que permiten trabajar con atributos continuos.

- La función objetivo ha de tener valores de salida discretos.
- Los datos de entrenamiento pueden poseer errores o ruido. Los árboles de decisión son lo suficientemente robustos si pueden trabajar con datos que contengan errores en sus atributos o en su clasificación.
- Puede ocurrir que falten valores para algunos de los atributos de los datos presentes en el conjunto de entrenamiento.

1.1.3. Redes Neuronales

En (Haykin, 1998) se define una 'red neuronal' como un procesador paralelo distribuido masivo, compuesto por unidades simples de procesamiento, las neuronas, que tienen una disposición natural para almacenar conocimiento producto de experiencias, y todo estará dispuesto para su uso.

En (Bishop, 2006) se consideran las redes neuronales como modelos eficientes para reconocimiento de patrones que pueden ser vistos de manera simple como una función no lineal que parte de un conjunto de variables x_i a un conjunto de variables de salida y_i; dicha función es controlada por un vector de pesos ajustables w.

1.1.4. Red *Feedforward*

La definición de una red *feedforward* según (Haykin, 1998) es una clase importante de red neuronal que típicamente consiste en un conjunto de unidades sensoriales que conforman la capa de entrada, una o más capas ocultas de neuronas computacionales y una capa de salida. La señal de entrada se propaga a través de la red en dirección de la capa de salida, pasando por cada capa intermedia.

1.1.5. *Backpropagation*

Backpropagation es una técnica específica para implementar el descenso de gradiente en los pesos de una red neuronal de tipo *feedforward*. En (Haykin, 1998),

la idea básica es calcular las derivadas parciales de una función aproximada *F(w,x)*. Esta función es obtenida a partir de la capa de salida de la red, con respecto a todos los elementos del vector de pesos w para un valor dado del vector de entrada x. Al obtener el valor asociado a la función, se propaga el error en dirección a la capa de entrada, partiendo desde la de salida y modificando el vector de pesos con la finalidad de minimizar el error de la red.

Por su parte, en (Mitchell, 1997) se define el *backpropagation* como un algoritmo que, dada una red neuronal con un conjunto fijo de unidades de procesamiento e interconexiones, aprende los pesos asociados con cada interconexión y aplica el método de descenso de gradiente con el objetivo de minimizar el error cuadrático entre la salida de la red y los valores deseados de salida.

1.2. Sistemas Multiagentes

"Un agente es un sistema de computadora que se encuentra ubicado en algún ambiente y es capaz de realizar acciones autónomas en ese entorno, con la finalidad de cumplir los objetivos que le han sido delegados."(Wooldridge, 2009).

En (Wooldridge, 2009) se añade además que los agentes deben ser reactivos, proactivos y con habilidades sociales. Reactivos porque los agentes deben ser capaces de percibir el ambiente en el que se desenvuelven y responder a los cambios que en él ocurren para satisfacer sus objetivos. Proactivos, ya que deben demostrar un comportamiento dirigido a sus objetivos, tomando la iniciativa de las acciones. Finalmente, deben tener habilidades sociales para lograr comunicarse con otros agentes. Si un agente cumple con esas tres condiciones es considerado un agente inteligente.

Existen tareas en las que la utilización de un solo agente para resolverlas es una limitante. Por eso hay sistemas multiagentes. Estos cuentan con al menos dos agentes que trabajan juntos para alcanzar un objetivo.

Los sistemas multiagentes pueden ser heterogéneos u homogéneos. Los heterogéneos tienen al menos dos agentes con diferentes capacidades de hardware o

software. Por otro lado, en los sistemas homogéneos todos los agentes son idénticos. Un ejemplo de un sistema multiagente heterogéneo son los robots utilizados en la Liga de Simulación de Fútbol 2D de la competición *RoboCup*, donde cada robot es programado para cumplir un rol diferente dentro del campo de juego (por ejemplo, arquero, defensa, mediocampista, delantero).

1.3. Prueba de hipótesis estadística

Una prueba de hipótesis estadística es definida por (Wackerly et al., 2002) como un procedimiento similar al método científico, en el cual se plantea una hipótesis respecto a los valores de uno o más parámetros poblacionales y en seguida se toma una muestra de la población y se comparan sus observaciones con la hipótesis. Si las observaciones no concuerdan con la hipótesis, se rechaza la hipótesis. De lo contrario, se concluye que la hipótesis es verdadera o que la muestra no detectó diferencia entre los valores reales e hipotéticos de los parámetros poblacionales utilizados.

1.3.1. Prueba t de Student

La prueba t de Student es descrita por (Berman y Wang, 2011) como una prueba que sirve para determinar si la diferencia de las medias de dos conjuntos es significativamente diferente de cero. Al realizar esta prueba, se toma como hipótesis nula que las medias de ambos grupos no presentan una diferencia estadísticamente significativa, y como hipótesis alternativa que sí. Para el uso de la prueba t para la comparación de las medias de dos grupos, que será la modalidad usada en esta investigación, se deben cumplir las siguientes condiciones: ambos grupos han de tener una distribución normal, ambos grupos deben poseer igual varianza y los datos usados para la realización de la prueba tienen que ser independientes los unos de los otros. Dado un nivel de significación α de 0,01, si el p-valor obtenido al realizar la prueba es inferior a α, la hipótesis nula es rechazada y se acepta la hipótesis alternativa; en caso de que el p-valor sea mayor a α, la hipótesis nula es aceptada.

1.4. RoboCup

RoboCup es una iniciativa científica internacional con la meta de avanzar en el estado del arte de robots inteligentes. Cuando se propuso en 1997, la misión original era la de desarrollar un equipo de robots capaz de ganar en un partido de fútbol en contra del equipo humano de fútbol que fuera campeón del mundo en el año 2050. Esta misión persiste a la fecha, pero *RoboCup* se ha expandido a otros dominios relevantes de aplicación, basándose en las necesidades de la sociedad actual (RoboCup, 2017). La competencia de *RoboCup* hoy en día está dividida en 4 categorías:

- RoboCup Soccer: esta es la categoría original para la que fue pensada la iniciativa de *RoboCup*. El enfoque principal de esta categoría, como su nombre lo indica, es el fútbol. El objetivo es impulsar la investigación en las áreas de sistemas multi-agentes que puedan cooperar en ambientes de confrontación. Todos los robots aquí utilizados deben ser completamente autónomos.
- RoboCup Rescue: esta categoría tiene como finalidad impulsar la investigación y el desarrollo de robots que puedan ser usados como asistencia a los servicios de emergencia en situaciones de desastres naturales para el rescate de víctimas.
- RoboCup@Home: en esta categoría se desarrollan robots de asistencia completamente autónomos para ayudar a las personas en tareas domésticas.
- RoboCup Junior: es un proyecto orientado a niños de primaria y secundaria que busca introducirlos al mundo de la robótica y darles los conocimientos necesarios dentro de las áreas de ingeniería, matemáticas y tecnología.

1.4.1. Liga de Simulación de Fútbol 2D

La Liga de Simulación de Fútbol en 2D es una de las subcategorías presentes en *RoboCup Soccer* y es, además, una de las más antiguas. Esta categoría deja a un lado la construcción física de los robots, pues esta puede generar diferencias notables entre los equipos, y se enfoca en mayor medida en el desarrollo de la Inteligencia Artificial y de las estrategias que quiera implementar cada equipo.

Esta competencia consiste en un partido de fútbol entre dos equipos de once agentes que juegan en un estadio virtual de dos dimensiones, representado por un servidor central llamado *SoccerServer*. El juego se basa en la comunicación entre el servidor y los agentes. Los jugadores reciben de forma individual información ruidosa referente al estado del juego por medio de sensores virtuales para luego con esta información ejecutar comandos básicos que influyen en su entorno, como por ejemplo: correr, voltearse, patear la pelota. (Federation, 2017)

1.5. Plataforma de Simulación

RoboCup Soccer Server es la plataforma oficial utilizada en la competencia *RoboCup* para la liga de simulación de fútbol en 2D. En esta investigación será utilizada la versión 15.2.2.

La plataforma es un sistema que posibilita a los agentes autónomos, escritos en diferentes lenguajes de programación, poder jugar un partido de fútbol. Para esto se emplea una arquitectura cliente-servidor.

En las arquitecturas cliente-servidor hay un *host* que se encuentra siempre activo, el servidor, el cual maneja las peticiones realizadas por otros *hosts*, llamados clientes(Kurose y Ross, 2012). En este tipo de arquitectura los clientes no se comunican entre sí, solo es necesario que puedan comunicarse cada uno con el servidor. Por esto, el servidor debe poseer una dirección IP fija, de tal modo que los clientes establezcan la comunicación enviando paquetes a esa dirección IP.

En este caso el servidor es el *SoccerServer* que contiene un campo de fútbol virtual y simula todos los movimientos permitidos a los jugadores y a la pelota. Del lado del cliente están los agentes que cumplen la función de los jugadores, encargados de controlar de forma individual sus propios movimientos. La comunicación entre los clientes y el servidor se realiza a través de sockets UDP/IP.(Chen et al., 2003)

La plataforma está compuesta por varios módulos que permiten simular cada aspecto del juego. Cada uno de ellos será explicado en las siguientes subsecciones.

1.5.1. SoccerServer

El servidor es un sistema en tiempo real que trabaja con intervalos de tiempo discretos, ciclos de 100ms. Cada tiempo del partido tiene una duración de 3.000 ciclos, aproximadamente 5 minutos. Como cada partido tiene al menos dos tiempos reglamentarios, su duración aproximada es de 10 minutos.

En caso de que algún jugador decida realizar alguna acción en un ciclo, el servidor es el encargado de aplicar la acción, en caso de estar permitida, en el siguiente ciclo. Luego de aplicar el cambio de posición de los objetos, el árbitro automatizado analiza la situación en la que se encuentran tanto los jugadores como la pelota de manera que se pueda realizar un cambio en el modo de juego de ser esto requerido. En caso de que este cambio se haya hecho, se informa inmediatamente a todos los agentes.

1.5.2. Soccer Monitor

Soccer Monitor provee una interfaz gráfica que le permite a los usuarios visualizar el juego en todo momento, además de otorgar la capacidad de controlar algunos aspectos del mismo. En el caso de la competición oficial de *RoboCup* existe un árbitro humano que puede hacer uso del monitor para penalizar acciones que el árbitro computarizado dejó pasar.

Además de las funcionalidades a la hora de competir, el monitor provee información útil para la etapa del desarrollo de los equipos. Es posible visualizar toda la información referente a cada jugador, según su tipo. También es posible cambiar el modo del juego en el que se encuentra el partido en un momento determinado.

La comunicación entre el *SoccerServer* y el monitor se realiza mediante *sockets* UDP/IP utilizando el puerto 6.000 por defecto. El servidor empleará este *socket* para enviarle información al cliente en cada ciclo del partido.

1.5.3. Soccer Client

Es el módulo correspondiente al manejo de los jugadores que tienen la función de clientes en la arquitectura cliente-servidor utilizada por la plataforma.

Cada jugador es capaz de ejecutar comandos que le posibilitan interactuar con el ambiente, como por ejemplo: girar el cuello o el cuerpo, correr, atrapar la pelota (solo para el arquero) y patear la pelota. Estos comandos son enviados por el agente que quiere ejecutarlos al servidor y éste los llevará a cabo al final de cada ciclo.

Además de realizar algunos movimientos físicos propios de un jugador de fútbol, los agentes poseen sensores que les permiten obtener información del estado del campo en determinado momento. Pueden escuchar mensajes que hayan sido enviados por sus compañeros, los jugadores del otro equipo, el árbitro o los entrenadores. También tienen la posibilidad de ver los objetos que se encuentran dentro de su rango de visión, identificados por tipo de objeto y la distancia a la que se encuentran.

Por último, cada jugador tiene un sensor en su cuerpo que le permite conocer su estado físico (velocidad, inclinación del cuello, esfuerzo realizado), así como también llevar contadores que posibilitan rastrear mensajes perdidos o que se encuentran retrasados como consecuencia natural del uso del protocolo UDP.

1.5.4. Coach

En *RoboCup Soccer* un *Coach* o entrenador es un cliente privilegiado utilizado para asistir a los jugadores, al igual que lo hace un entrenador en el fútbol convencional. Este cliente puede comunicarse con los jugadores y recibir información libre de ruidos del estado del juego en todo momento.

El entrenador es el encargado de realizar las sustituciones de los jugadores. Antes del inicio del partido puede hacer cambios ilimitados en el tipo de los jugadores. Luego de iniciado el partido, puede realizar máximo tres reemplazos siempre que el modo de juego que esté en el momento del envío del mensaje no sea "*play_on*".

Además, el entrenador puede enviar mensajes libres a los jugadores (máximo 128 caracteres) que son usados para informarles de situaciones importantes dentro del campo de juego, cambiar las formaciones del equipo, generar estrategias, etc. Para evitar que el entrenador aplique un control centralizado del comportamiento de los jugadores, el servidor impone unas restricciones en el envío de estos mensajes. Por esto, si el modo de juego es "*play_on*", estos mensajes solo pueden ser enviados cada 600 ciclos durante 20 ciclos. En caso de que exista empate y sea necesario jugar tiempo extra, se aumenta la cantidad de mensajes que puede enviar el entrenador durante un partido.

Es importante destacar que *SoccerServer* garantiza que los mensajes serán enviados a los jugadores en el mismo orden en el que fueron recibidos del entrenador.

1.5.5. Logs

Entre las funciones que presenta el servidor *SoccerServer* está la de permitir llevar un registro de todo lo ocurrido durante un partido. Al finalizar el juego el servidor genera dos *logfiles* o archivos de registro, que contienen toda la información, sin ruido, de lo realizado por cada uno de los agentes de ambos equipos.

Combinados con la funcionalidad del monitor, estos datos pueden ser utilizados para reproducir el partido como si se estuviera jugando en tiempo real. Además, los datos que se encuentran en estos archivos pueden ser empleados para analizar el comportamiento y las estrategias de los equipos, así como también descubrir sus ventajas y fortalezas en determinadas situaciones y de este modo generar estrategias en el equipo que los analiza para aprovechar la información extraída.

1.6. Equipo base Agent2D

Agent2D Base es un equipo base para la liga de simulación de fútbol 2D de RoboCup implementado por el equipo japonés Helios. Se desarrolló en *Standard C++* y es soportado en distintos sistemas operativos como Linux, Mac OSX y Windows. Aunque es un código con suficiente complejidad para que los agentes

trabajen como un equipo, es suficientemente simple para ser empleado como base para la creación de estrategias y comportamientos avanzados (Behnke et al., 2014).

La influencia de este *software* dentro de la liga de simulación 2D queda en evidencia con los datos presentados en la Tabla 1.1. Allí se puede observar la cantidad de equipos participantes en la categoría de simulación de fútbol 2D de RoboCup entre los años 2007 y 2013, junto al número de estos equipos cuyo código fuente estaba basado en el de *Agent2D Base*, siendo éste el equipo más usado desde el año 2012.

Año	# de equipos participantes	# de equipos Agent2D Base	Porcentaje
2007	15	3	20
2008	15	4	26,7
2009	19	7	36,5
2010	19	8	42,1
2011	17	8	47,1
2012	18	15	83,3
2013	24	20	83,3

Tabla 1.1: Transición del número de equipos que usan Agent2D Base

El uso de equipos base para comenzar a desarrollar nuevas funcionalidades ha existido desde hace muchos años en RoboCup. Antes de que el código fuente de *Agent2D Base* se compartiera de forma pública, el equipo base más utilizado era *UvA Trilearn*, desarrollado por Jelle Kok y Remco de Boer en la Universidad de Amsterdam (de Boer y Kok, 2002).

Los equipos prefieren centrar su trabajo en la elaboración de estrategias y no en la implementación de funcionalidades básicas para la interacción de los agentes, como crear una red estable de comunicación, la sincronización entre los agentes y el modelo del mundo en el que interactúan. Al tener estos módulos disponibles en un equipo base, los esfuerzos pueden centrarse en la esencia de la categoría de simulación de RoboCup: los sistemas multi-agentes.

Capítulo 2

Diseño e implementación del coach

La implementación del coach para el equipo base se divide en tres grandes funcionalidades enfocadas en distintos aspectos: la detección de acciones durante el partido, el reconocimiento del esquema táctico de los equipos oponentes y la adaptación de la formación del equipo base para contrarrestar la estrategia del oponente.

Se busca que al utilizar alguna de estas nuevas destrezas se mejore el desempeño del equipo *Agent2D Base*. Como se dijo, este equipo es bastante simple, pues cuenta con las habilidades necesarias para enfrentarse a otros equipos, pero posee estrategias poco complejas que dificultan la tarea de derrotarlos.

El equipo *Agent2D Base* original cuenta con un *coach* que realiza labores básicas de un director técnico de fútbol. La primera tarea que cumple es la de seleccionar los jugadores que estarán en el once inicial del equipo, jugadores heterogéneos con características diferentes según los tipos disponibles. Cuando estos cambios de tipo de jugador se realizan después de iniciado el juego, solo podrán hacerse con tres sustituciones si el partido no termina en empate durante el tiempo reglamentario.

Además, durante los juegos el *coach* busca identificar los tipos de cada jugador del equipo oponente para enviarle esta información a sus jugadores, y que tengan así conocimiento del ambiente en el que se están desenvolviendo.

Desde ahora se hará la distinción entre el equipo original, *Agent2D Base*, que servirá como base de esta investigación, y el equipo modificado con una o varias de las nuevas funcionalidades, al cual se hará referencia como el equipo JEMV.

En las siguientes secciones se explicará en profundidad las bases de cada una de las funcionalidades que serán agregadas al coach de *Agent2D Base*, además de los detalles de su implementación.

2.1. Selección de equipos oponentes

La escogencia de los equipos oponentes que se utilizarán para evaluar el desempeño de las variaciones de JEMV es un factor de gran importancia para esta investigación. Seleccionar solo equipos campeones puede hacer difícil la detección de algún tipo de mejora por parte de los jugadores del equipo que se está desarrollando.

Por otro lado, basar la significancia de la mejora de JEMV en sus enfrentamientos contra solo equipos de baja dificultad no permite realizar un análisis completo del impacto real de las destrezas que se están incorporando tanto a los jugadores como al *coach*.

Además de tomar en cuenta el nivel de dificultad de los oponentes, se buscó que los seleccionados fueran equipos actuales, que hayan participado al menos hasta el año 2015 en la liga de RoboCup. Tomando en cuenta las dos características mencionadas anteriormente, se eligieron cinco equipos actuales, con distintos niveles de dificultad.

El equipo Wright Eagle es el perfil de un equipo campeón en el que han trabajado casi desde los inicios de la categoría simulada de RoboCup. Está siendo desarrollado por el Laboratorio de Sistemas Multi-agente de la Universidad de Ciencia y Tecnología de China (USTC) y ha participado en las competiciones anuales de RoboCup desde el año 1999. Además, es el equipo que ha conseguido el primer lugar en más ocasiones en toda la historia de la categoría de simulación 2D, y fueron campeones en seis años (2006, 2009, 2011, 2013, 2014 y 2015). En los años

2005, 2007, 2008, 2010 y 2012 consiguieron el segundo lugar de la competencia. (Li et al., 2015)

Además de Wright Eagle, fue seleccionado otro equipo campeón de RoboCup, el conjunto japonés Helios (Akiyama et al., 2016). Helios es desarrollado por un equipo de investigadores de la Universidad de Fukuoka, en Fukuoka, Japón. Han conquistado el primer lugar de la competición en dos ocasiones (2010 y 2012), consiguiendo además el segundo lugar en cuatro años (2009, 2011, 2013 y 2015), el tercero de forma consecutiva en los años 2007 y 2008, y el cuarto lugar en el año 2014.

La importancia de Helios no solo queda en evidencia con los títulos obtenidos y los avances que han aportado con las mejoras de su equipo, sino que también son los creadores del equipo base más utilizado actualmente dentro de la competencia que, como fue mencionado en el capítulo anterior, será utilizado como base de esta investigación.

El tercer equipo que se empleará para la evaluación de las nuevas habilidades del equipo JEMV es Hermes. Este equipo está basado en *Agent2D Base* y es desarrollado por el Departamento de Ciencias de la Computación y Tecnología de Información de la escuela secundaria Allameh Helli, en Teherán, Irán (Javan et al., 2016). A diferencia de los dos equipos anteriores, Hermes no ha conseguido alcanzar ninguno de los primeros lugares en los tres años en los que ha participado (2016, 2015 y 2014); lo anterior tiene explicación si consideramos que es un equipo nuevo que solo ha desarrollado estrategias básicas ofensivas y defensivas para sus jugadores.

Finalmente, los últimos equipos elegidos fueron Genius y Jaeger. Ambos han sido desarrollados a partir de *Agent2D Base*, y han participado en varias ocasiones dentro del mundial de RoboCup y los abiertos de Irán y China. Jaeger es implementado por un equipo de la Universidad de Tecnología de Anhui, en China, y Genius fue desarrollado por la escuela secundaria Ghazal (Shiraz) en Irán.

2.2. Detección de acciones durante el partido

Una forma de mejorar el desempeño del equipo *Agent2D Base* es entrenar a sus jugadores para que sean capaces de discernir si es conveniente realizar una acción en un momento dado del partido, dependiendo del estado del ambiente en ese instante. Para lograr esto es necesario extraer información relevante de los *logs* generados por el servidor, producto de partidos anteriores entre el equipo JEMV y el oponente al que se debe enfrentar.

Esta funcionalidad del *coach* está basada en el trabajo de (Karimi y Ahmazadeh, 2014), quienes realizan la extracción de acciones previo a los partidos y la información obtenida es utilizada únicamente por los jugadores en tiempo real. Se pensó conveniente agregar la figura del *coach* a esta idea, para permitir el entrenamiento de los modelos de aprendizaje de máquina durante el transcurso de los juegos. Además, se realizaron cambios en la forma de calcular los atributos de los árboles de decisión, como se explicará más adelante en este capítulo.

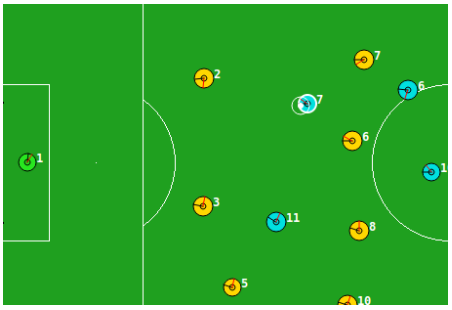
Las acciones que se extraerán de los *logs* corresponden a los movimientos básicos del fútbol real: dribles, pases y tiros al arco. Además, cada una de ellas se identificará como una acción exitosa o fallida según el caso que corresponda.

Para entender el proceso de extracción de acciones realizado es preciso definir el concepto de dueño de la pelota en un ciclo X del juego. Este será el jugador de cualquiera de los dos equipos que se encuentre más cerca del balón en el ciclo X, solo si la distancia entre este jugador y la pelota es menor a 0.05 unidades de distancia. Si para ese momento ningún jugador cumple con esta condición, se considerará como dueño al valor nulo.

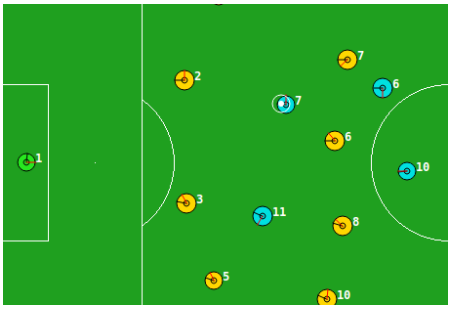
Se define una acción como un drible cuando el jugador dueño de la pelota en un ciclo X, se mantiene en su posesión durante varios ciclos posteriores. Si un oponente del dueño del balón intercepta la acción a una distancia menor a 5 unidades, generando un cambio en la posesión de la pelota, se considerará como un drible fallido.

En la figura 2.1 se ejemplifica una posible ejecución de un drible durante un

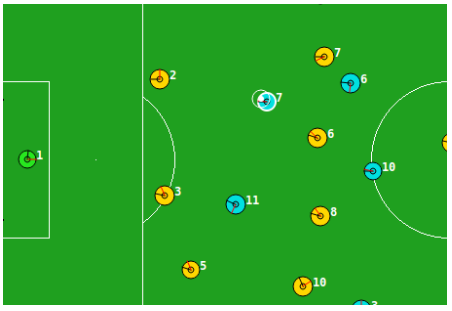
partido de la Liga Simulada de RoboCup. Inicialmente, en la figura 2.1(a) el jugador número 7 del equipo azul es considerado el dueño de la pelota, que está representada por un círculo interno blanco relleno y uno externo sin fondo. Varios ciclos después el estado del juego se encuentra como en la figura 2.1(b), continúa la acción del drible, pero el jugador no ha tomado posesión nuevamente de la pelota, esto es fácilmente identificable puesto que la silueta del jugador no está resaltada con color blanco. El drible finaliza en la figura 2.1(c) cuando el dueño de la pelota vuelve a ser el jugador 7 del equipo azul.



(a) Inicio del drible



(b) Drible en proceso



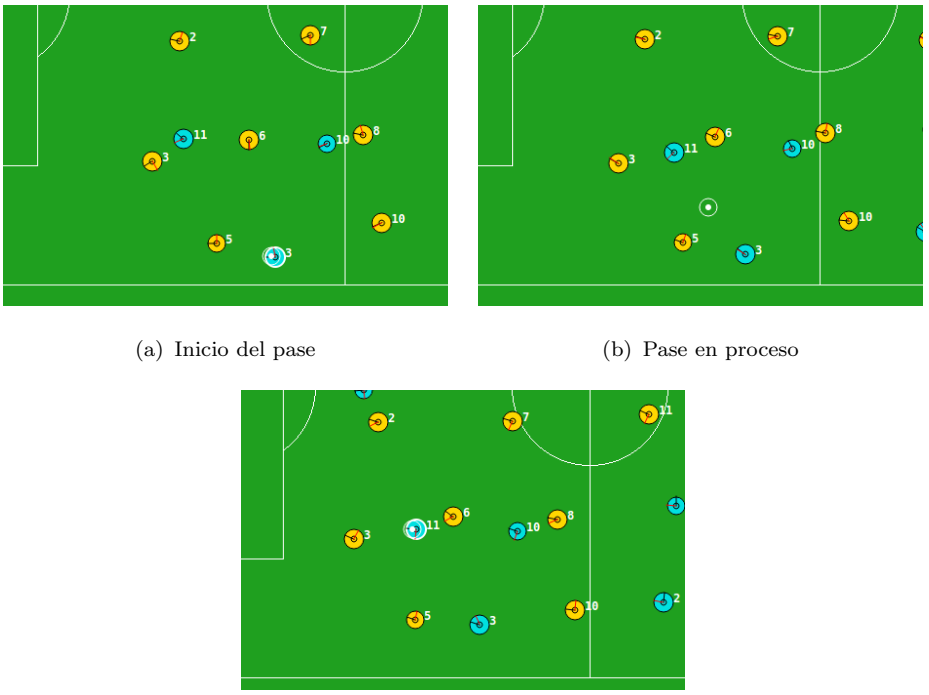
(c) Fin del drible

FIGURA 2.1: Ejemplo de la ejecución de un drible durante un partido de RoboCup

Un pase es exitoso si el jugador patea el balón y otro agente del mismo equipo lo recibe, convirtiéndose en el nuevo dueño. Se considera fallido si el balón es interceptado por un jugador del equipo contrario a una distancia mayor o igual a 5 unidades de distancia del dueño.

La figura 2.2 ilustra el proceso de un pase dentro de un partido de RoboCup. En la figura 2.2(a) el dueño de la pelota es el jugador 3 del equipo azul, este es considerado el instante inicial del pase. Ciclos después, el mundo se encuentra como

en la figura 2.2(b), donde no existe un jugador dueño de la pelota. Finalmente, en el ciclo correspondiente a la figura 2.2(c) el jugador 11 del equipo azul recibe la pelota, y se convierte en el nuevo dueño, dando fin a la acción del pase.



(a) Inicio del pase

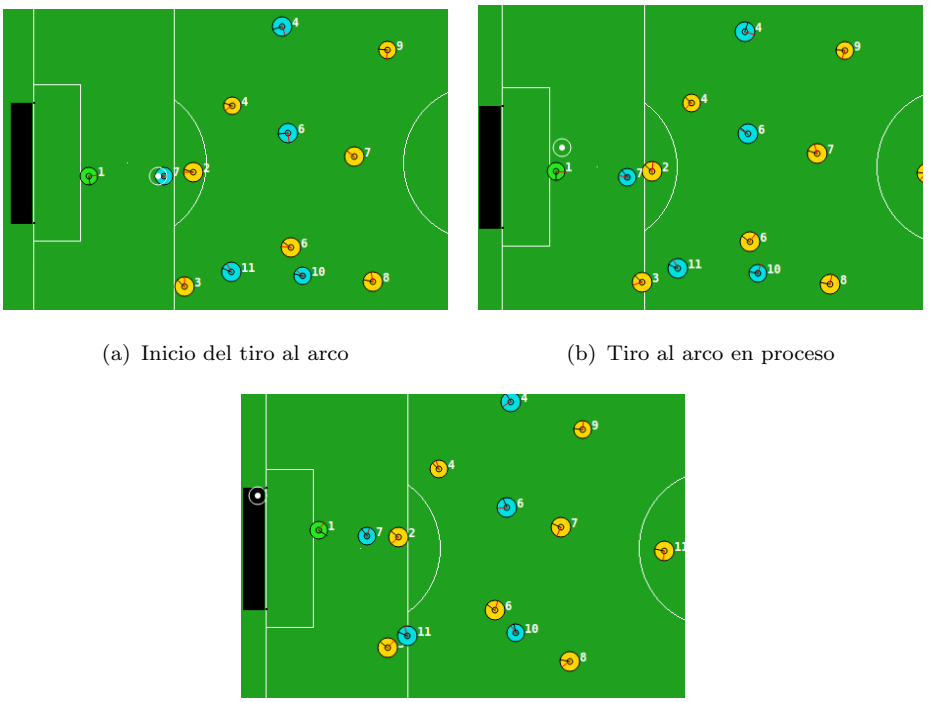
(b) Pase en proceso

(c) Fin del pase

FIGURA 2.2: Ejemplo de la ejecución de un pase durante un partido de RoboCup

Los tiros al arco se consideran exitosos si el balón entra al arco, anotando un gol, y se consideran fallidos si el balón está moviéndose en dirección al arco y este es interceptado por un oponente en un área cercana.

En la figura 2.3 se ve un ejemplo de la ejecución de un tiro al arco. Inicialmente, el dueño de la pelota es el jugador 7 del equipo azul y este constituye el instante inicial de la acción. En la figura 2.3(b) el jugador 7 ya ha disparado la pelota en dirección a la portería contraria. Por último, en el ciclo de la figura 2.3(c) la pelota ingresa a la portería del equipo amarillo, anota un gol y se da por finalizada la acción.



(a) Inicio del tiro al arco

(b) Tiro al arco en proceso

(c) Fin del tiro al arco

FIGURA 2.3: Ejemplo de la ejecución de un tiro al arco durante un partido de RoboCup

Teniendo las acciones definidas, el siguiente paso es analizar partidos anteriores generados entre el equipo JEMV y el oponente que se quiere estudiar. Para esto se utiliza la información provista por el *SoccerServer* a través de los *logs*. En cada ciclo se identifica cuál jugador es el dueño de la pelota y, basados en las condiciones planteadas anteriormente, se reconocerá la acción respectiva.

Una vez extraídas las acciones para cada partido, deben determinarse los valores que serán empleados como atributos de los modelos de aprendizaje de máquina que se seleccionen. En el trabajo original en el que se basa la implementación de esta destreza del *coach*, los autores realizan dos preprocesamientos de los datos para generar los atributos finales (Karimi y Ahmazadeh, 2014).

El primer proceso busca reducir la dimensionalidad de los datos con la finalidad de generar un modelo más rápido y efectivo. Bajo la suposición de que no todos los jugadores dentro del campo participan de forma activa dentro de cada acción, Karimi y Ahmazadeh decidieron escoger solo 7 jugadores del total de 22.

Además de las coordenadas de la posición de la pelota, escogen a tres jugadores que forman parte del equipo que se encuentra en posesión de ella: el dueño de la pelota, el compañero ubicado más cerca del dueño de la pelota y el compañero más cercano al punto final de la pelota en esa acción. El resto de los jugadores se seleccionan del equipo oponente: el oponente más cercano a la posición inicial del balón, el más cercano al punto final y dos oponentes cercanos al centro de la trayectoria realizada por la pelota durante la acción.

El segundo preprocesamiento que realizan sobre los datos es el que se muestra en la Tabla 2.1. Además, se dividen los conjuntos de datos por clase. Los datos asociados a las clases 'Drible' y 'Drible fallido' servirán para la creación del modelo clasificador de dribles, los de las clases 'Pase' y 'Pase fallido' para el modelo de pases y, finalmente, los de las clases 'Tiro al arco' y 'Tiro al arco fallido' para el modelo de tiros al arco.

Atributo	Definición
Pelota.x, Pelota.y	Coordenada x/5 y coordenada y/5
Compañero x(1,2,3)	Distancia entre el compañero x y la pelota
Oponente z(1,2,3,4)	Distancia entre el oponente z y el x más cercano a él
Acción	Identificador de la clase de la acción asociada al modelo

Tabla 2.1: Atributos originales de los árboles de decisión del trabajo de Karimi y Ahmazadeh (2014)

Los resultados obtenidos por Karimi y Ahmazadeh no pudieron ser replicados en esta investigación usando árboles de decisión; esto se debió, tal vez, por la suposición de procesos que debieron hacerse sin información completa en algunos aspectos de su trabajo. A pesar de ello, se decidió continuar con la idea base de la extracción de las acciones y la definición de cada una de ellas utilizada por dichos autores. Los cambios se realizaron en la cantidad de atributos seleccionados para los árboles y su definición.

Los atributos asociados a la posición de la pelota se mantuvieron dentro de los elegidos para los modelos. Se realizaron pruebas incluyendo otras variables físicas de la pelota, como su velocidad, pero no agregaban información significativa y empeoraban el desempeño de las predicciones.

Los cambios principales que se hicieron a los atributos originales giran en torno de las siguientes interrogantes sobre los jugadores de ambos equipos: qué información de los jugadores es útil para los modelos y cuáles jugadores serán seleccionados.

La primera pregunta fue resuelta con base en la idea de que, al momento de realizar una acción, la prioridad reside en que ningún jugador oponente debe estar presente dentro de la trayectoria que recorrerá la pelota o, al menos, que algún compañero esté más cerca de la misma que cualquier oponente. Con esto se aseguraría que si el jugador destino no recibe la pelota, al menos otro de los compañeros podrá interceptarla sin perder la posesión.

Por esta razón se decidió que los atributos estarían definidos por la distancia entre la posición de los jugadores seleccionados al momento de iniciarse la acción y por la recta que se forma al unir las coordenadas de la posición de la pelota en el instante inicial y en el final.

La segunda interrogante, cuáles jugadores seleccionar, fue respondida a partir de los experimentos. Se evaluaron los modelos partiendo de la suposición de que todos los jugadores afectan de alguna manera la ejecución de una acción, bien sea de forma activa, siendo alguno de los actores directos de la jugada, o de forma pasiva, arrastrando la posición de los oponentes como forma de distracción.

Partiendo de dicho modelo se intentó encontrar el mínimo número de jugadores que fueran necesarios para mantener o superar los resultados obtenidos con las predicciones de los árboles de decisión que manejaban las distancias de los 22 jugadores y la posición de la pelota, seleccionando siempre a los que estuvieran más cerca de su trayectoria.

Tras estos experimentos se concluyó que el mejor modelo es el obtenido con los 22 jugadores, cuyos resultados serán presentados en el Capítulo 3. Se cree que tal resultado es debido a la poca complejidad de las estrategias del equipo *Agent2D Base* que ocasionan que en gran parte del partido las líneas de juego (defensa, mediocampo y ataque) se encuentren cercanas intentando defenderse del ataque oponente.

2.2.1. Detalles del modelo modificado

Luego de conocer el proceso aplicado para la construcción del modelo original del trabajo de (Karimi y Ahmazadeh, 2014), en esta subsección se explicará de forma específica cuáles fueron las modificaciones necesarias para la elaboración de los modelos de árboles de decisión finales para la clasificación de cada tipo de acción; desde la selección de los datos, pasando por los atributos seleccionados para modelar cada problema, hasta el algoritmo utilizado por la librería OpenCV para la construcción de los árboles.

El primer paso para el diseño de la solución es el de la selección de los datos. Para obtener los datos necesarios utilizados en el entrenamiento de los árboles, se generaron 350 partidos entre cada uno de los equipos oponentes elegidos y el equipo *Agent2D Base*. Sin embargo, la cantidad de acciones que se obtuvo para cada conjunto de datos separado por equipos es diferente, puesto que tales acciones dependen del comportamiento que tengan los jugadores durante cada partido, aspecto que no puede ser controlado.

Es importante destacar que las acciones empleadas para la construcción de los conjuntos de datos son únicamente las del equipo *Agent2D Base*, puesto que las habilidades de los jugadores del equipo base son distintas de las de los cinco equipos oponentes, y hacer uso de las acciones que ellos realizan simplemente para predecir y no para imitar llevaría a resultados erróneos en los modelos.

Una vez extraídas las acciones de los partidos de cada oponente, se divide nuevamente el conjunto de datos en tres nuevos grupos: uno que contiene solo acciones de tipo ‘Drible’ y ‘Drible fallido’ para la clasificación de dribles, otro con datos de ‘Pase’ y ‘Pase fallido’ para la clasificación de pases, y finalmente un conjunto con los elementos de las clases ‘Tiro al arco’ y ‘Tiro al arco fallido’. Esto da como resultado tres árboles de decisión para cada equipo oponente.

Los atributos utilizados para los tres árboles de cada equipo son los presentados en la Tabla 2.2. Es importante destacar que las posiciones tomadas en cuenta tanto para los primeros dos atributos de la pelota como para los jugadores corresponden a las posiciones en el tiempo inicial de la acción. Este aspecto no se encontraba explícito en el trabajo original (Karimi y Ahmazadeh, 2014), pero debido a que los

atributos deben extraerse también en tiempo real antes de que la acción ocurra, este sería el único instante en el que se tienen valores certeros de las posiciones.

Atributo	Definición
Pelota.x, Pelota.y	Coordenada x/5 y coordenada y/5
Compañero x_i, i=1..11	Distancia entre la posición del compañero x_i al inicio de la acción y la recta formada por las posiciones inicial y final de la pelota
Oponente z_j, j=1..11	Distancia entre la posición del oponente z_j al inicio de la acción y la recta formada por las posiciones inicial y final de la pelota
Acción	Identificador de la clase de la acción asociada al modelo

Tabla 2.2: Atributos finales utilizados para los árboles de decisión del equipo JEMV

Todos los atributos utilizados corresponden a variables numéricas, de las cuales siempre se conoce su valor. El atributo correspondiente a la clase o etiqueta de cada dato es un valor categórico al que se le asigna '1' si la clase corresponde a un valor positivo ('Drible', 'Pase' o 'Tiro al arco'), y '-1' si pertenece a un valor negativo ('Drible fallido', 'Pase fallido' o 'Tiro al arco fallido').

Luego de generar los conjuntos de datos iniciales para cada tipo de acción, se realizó un preprocesamiento para mejorar la calidad de estos antes de ser presentados a los árboles. El preprocesamiento consiste en una normalización que permite escalar los atributos dentro del rango [0,1], buscando así mejorar la precisión de las predicciones. Esta normalización se llevó a cabo a partir de la siguiente ecuación para cada uno de los atributos numéricos:

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{2.1}$$

donde x representa el valor original, x' es el valor normalizado, $min(x)$ es el mínimo valor existente en el rango original del atributo al que pertenece x, y $max(x)$ el límite superior de dicho rango. Los mínimos y máximos de cada rango

identificados en la fase de entrenamiento serán también utilizados en tiempo real durante la ejecución de los partidos.

Este proceso de normalización generó mejores resultados tanto en los árboles de decisión como en las redes neuronales presentadas en el Apéndice B. Esta técnica permite evitar la saturación u *overflow* de las neuronas de las redes, además de ayudar a visualizar y manipular de mejor forma los datos, y facilitar la comparación de resultados con otros modelos.

Luego de generar todos los conjuntos de datos, fue necesario pre-entrenar árboles de decisión para cada tipo de acción y separarlos por equipos oponentes. Antes de realizar un drible, pase o tiro al arco, los jugadores deben cargar el árbol de decisión asociado a la acción que desean ejecutar y al equipo oponente al que se enfrentan. Luego, calculan los valores de los atributos respecto al estado del mundo en el instante en el que se encuentran. La posición final de la pelota viene estimada por la posición del jugador objetivo al inicio de la acción.

Una vez calculados todos los atributos necesarios, el jugador utiliza el árbol de decisión para ejecutar la predicción y determinar si la acción que desea efectuar será beneficiosa para su equipo. En el caso de que el árbol retorne una predicción positiva, el jugador procede a ejecutar la acción. Si la predicción es negativa, se evita esa jugada y se continúa con la siguiente dentro de la cadena de acciones que posee el dueño de la pelota. De suceder que ninguna otra jugada es posible, o todas fueron rechazadas, el jugador retendrá la pelota hasta el siguiente ciclo cuando la cadena de acciones arroje nuevos resultados.

La participación del *coach* en este proceso ocurre en tiempo real durante la ejecución del partido. Este agente se encarga de monitorear el desenvolvimiento del partido, almacenando todas la acciones ejecutadas por el equipo JEMV y su resultado, fueran exitosas o no. Esto permitirá entrenar nuevamente los árboles de decisión del oponente, tomando los datos originales junto a los acumulados por el *coach* en tiempo real.

La decisión de entrenar nuevamente los árboles al momento de recibir un gol fue tomada porque en ese instante cambia el modo de juego, dejando de ser *PlayOn*, y los jugadores no necesitan hacer consultas sobre el árbol de decisión. Con esto se busca abarcar tanto los comportamientos pasados del oponente como un posible

cambio en sus reacciones dentro del partido actual.

Los árboles de decisión empleados para el diseño de estos modelos fueron los de la librería de *Machine Learning* de OpenCV. Los árboles de esta librería son implementados con el algoritmo CART descrito en (Breiman et al., 1984).

2.3. Reconocimiento de formaciones del equipo oponente

El *coach* implementado tiene la capacidad de reconocer la formación del equipo oponente y, para ello, emplea tanto la experiencia extraída de partidos anteriores como en tiempo real durante la ejecución de un juego. Una vez reconocida la formación empleada por el equipo oponente, se usará esta información para contrarrestar su estrategia, lo que modificará el esquema de juego del equipo JEMV.

El método empleado para reconocer la formación de un equipo hace uso de las posiciones relativas de los jugadores en el campo, de esta forma el proceso se vuelve generalizado puesto que no es afectado por el lugar en el espacio donde se esté desarrollando el partido al momento de verificar las posiciones.

El campo de juego tiene una dimensión de 104x68 unidades; se ubica el centro en el punto (0,0) y las cuatro esquinas del rectángulo en (-52,34), (52,34), (-52,-34) y (52,-34), tal como se muestra en la figura 2.4.

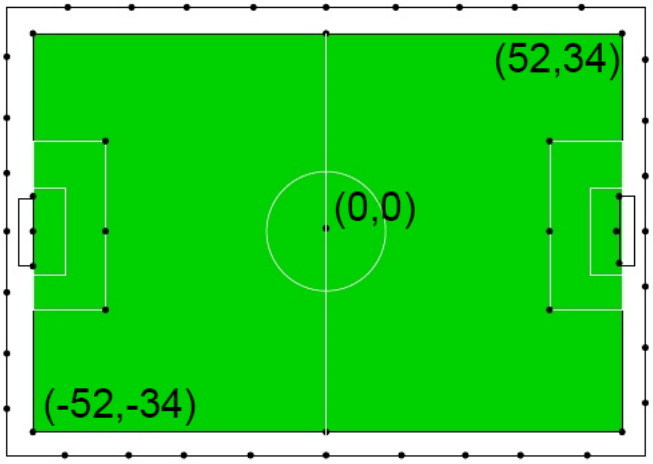


FIGURA 2.4: Campo de juego.

El primer paso para el reconocimiento de la formación consiste en realizar un preprocesamiento del campo de juego; este es segmentado en áreas rectangulares de tamaño 2,97x2 unidades cada una. Se eligió esta área para cada rectángulo ya que los jugadores tienen un tamaño de 0,3 unidades cada uno, y esto permite tener un recuadro suficientemente pequeño para que no se pierda información respecto a los movimientos de un jugador al trasladarse y suficientemente grande para que sean necesarios varios movimientos cortos, o un movimiento brusco para trasladar a un jugador a la siguiente área.

La segmentación del terreno permite generar matrices de dimensión 35x34 para cada uno de los jugadores del equipo que será analizado, matrices que servirán para representar en qué áreas del campo se encuentra cada jugador durante los ciclos que han transcurrido. Es importante destacar que este algoritmo solo tomará en cuenta la posición de los jugadores de campo, es decir, el guardameta del equipo que está siendo analizado no es tomado en cuenta dentro del cálculo de las matrices ni en el análisis de la formación. Esto se debe a que su posición en el campo no influye en el esquema táctico del equipo, pues el área donde se desenvuelve en situaciones normales es conocida

Este procedimiento se basa en el trabajo realizado en (Habibi et al., 2002), con la diferencia de que el reconocimiento ahí planteado utiliza una sola matriz general en la que se incluye la información de los 10 jugadores que se encuentran en el campo. A continuación se mostrarán los resultados de ambos métodos aplicados al equipo Hermes para obtener su formación después de 50 partidos.

Las figuras 2.5 y 2.6 fueron extraídas de dos partidos distintos de Hermes contra el equipo *Agent2D Base*, como ejemplos gráficos del total de 50 partidos empleados para evaluar ambos métodos. En estas dos imágenes podemos observar que el equipo Hermes (equipo azul, del lado derecho) utiliza una formación 4-3-3 (4 defensas, 3 mediocampistas y 3 delanteros).

FIGURA 2.5: Imagen de Hermes en juego. (Equipo azul, lado derecho)

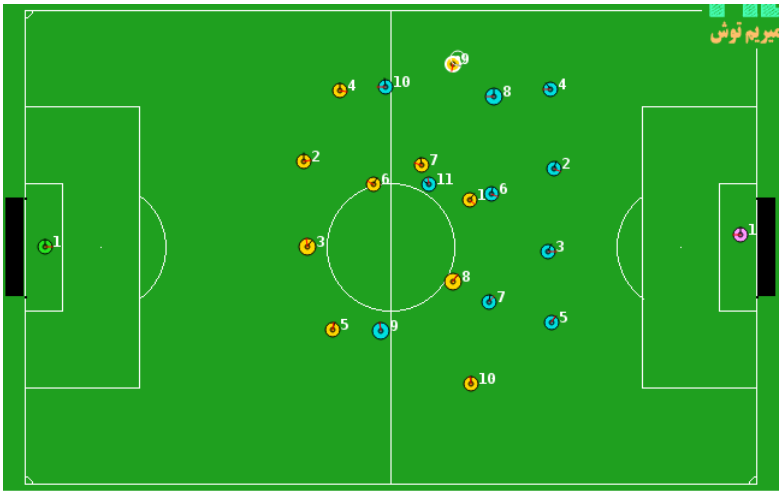


FIGURA 2.6: Imagen de Hermes en juego. (Equipo azul, lado derecho)

En las figuras 2.7 y 2.8 se muestran los resultados del proceso; se usaron 1 matriz y 10 matrices, respectivamente. En las imágenes, cada punto representa una de las áreas en las que se dividió el campo inicialmente; los tonos más oscuros representan áreas de mayor actividad para los jugadores, mientras que los más claros representan menor actividad. Cabe destacar que los resultados mostrados suponen que el equipo oponente juega del lado izquierdo, de manera que los puntos representan a los jugadores en posiciones de defensa a la izquierda y delanteros a la derecha.

La figura 2.7 muestra la actividad general del equipo en el campo, con las áreas oscuras que indican mayor actividad, pero determinan que la detección de una

formación clara sea difícil, ya que no es posible diferenciar entre la actividad de cada jugador en específico.

La figura 2.8 muestra el resultado de sobreponer la matriz de cada jugador en una sola imagen y permite que se observe de manera clara una formación 4-3-3 en el campo en las áreas de mayor actividad para cada jugador. Algunos tonos grises representan el movimiento de los jugadores en dichas áreas.

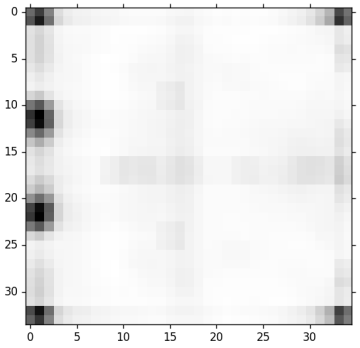


FIGURA 2.7: Resultado con 1 matriz.

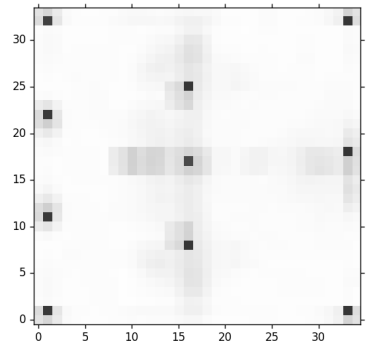


FIGURA 2.8: Resultado con 10 matrices.

Debido a la forma en la que se calcula la actividad de los jugadores, hacer uso de una sola matriz causa tonos muy oscuros en áreas donde muchos jugadores pasaron tiempo, mientras que sitios donde solo un jugador pasó mucho tiempo presentan un tono más gris, haciendo difícil el proceso de identificar una formación.

Al hacer uso de 10 matrices, es posible conocer el área exacta donde cada jugador pasó la mayor cantidad de tiempo y se puede emplear esta como su lugar en la formación, mientras que, usando una matriz, solamente es posible diferenciar donde hubo mucha actividad, pero no por parte de cuál jugador. Por ello en esta investigación se hará uso del método planteado de 10 matrices que detectan formaciones.

Luego de haber generado las matrices para los 10 jugadores del equipo que se quiere analizar se procede a procesar los datos del partido ciclo a ciclo. En cada uno se determina el rectángulo de área mínima que pueda cubrir a todos los jugadores oponentes que se encuentren en el campo, tomando sus posiciones en ese instante. Este rectángulo es empleado para trasladar las coordenadas de cada jugador al campo de referencia inicial segmentado usando las siguientes fórmulas:

$$xNuevo = \frac{x - origenX}{anchoRect} * anchoRef - \frac{anchoRef}{2} \tag{2.2}$$

$$yNuevo = \frac{y - origenY}{altoRect} * altoRef + \frac{altoRef}{2} \tag{2.3}$$

donde ($xNuevo$, $yNuevo$) será la posición del jugador respecto al campo de referencia; (x,y) es la posición original; ($origenX$, $origenY$) es la ubicación de la esquina superior izquierda del rectángulo mínimo, que es empleado como su origen; $altoRect$, $anchoRect$, $altoRef$ y $altoRect$ representan las dimensiones del rectángulo mínimo y del campo de referencia, respectivamente.

En la figura 2.9 se muestra un ejemplo de la obtención del rectángulo mínimo que cubre al equipo oponente (equipo amarillo, lado izquierdo). En la figura, las líneas blancas que dividen el campo no son las divisiones de 2,97x2 mecionadas anteriormente, pues forman solo una cuadrícula para hacer más fácil la visualización de las posiciones y su transformación al campo de referencia.

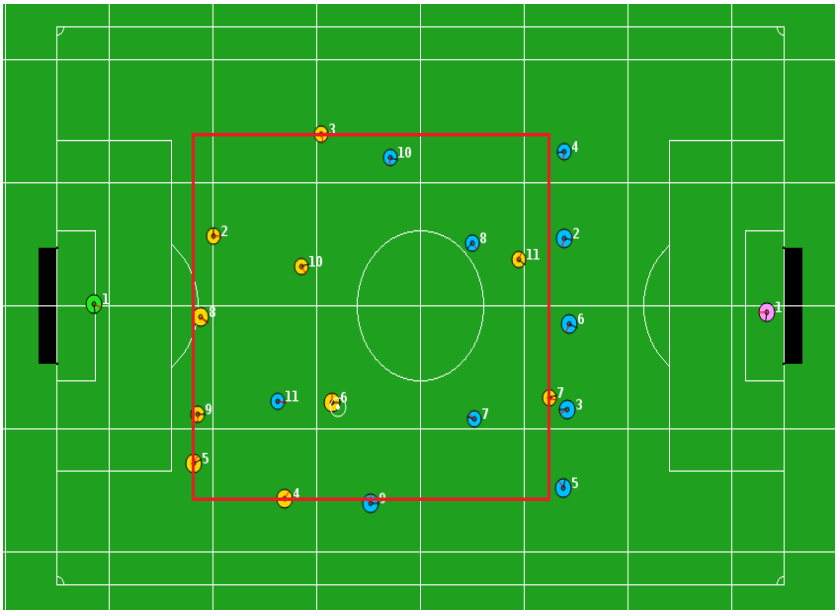


FIGURA 2.9: Rectángulo mínimo que cubre al equipo oponente.

Teniendo las posiciones relativas de cada oponente, se verifica para cada uno de ellos en qué rectángulo del campo de referencia segmentado previamente fue ubicado luego de la transformación de sus coordenadas, aumentando en 1 en la posición de ese rectángulo dentro de su matriz asignada. La transformación de coordenadas mencionada se puede ver en la figura 2.10 en donde los rectángulos de borde negro

representan las áreas de tamaño 2,97x2 y las líneas amarillas muestran el traslado de la posición real del jugador al campo de referencia en el área rectángular correspondiente. Adicionalmente se aumenta en 1 también los vecinos directos del rectángulo (arriba, abajo, izquierda, derecha y diagonales) para agregar la noción de vecindad y no limitar al jugador a una posición exacta, puesto que no son entes estáticos. Este proceso es repetido en cada ciclo del partido. Finalmente se obtiene una matriz de enteros para cada jugador que indica el tiempo que pasaron en cada área de los rectángulos usados como referencias.

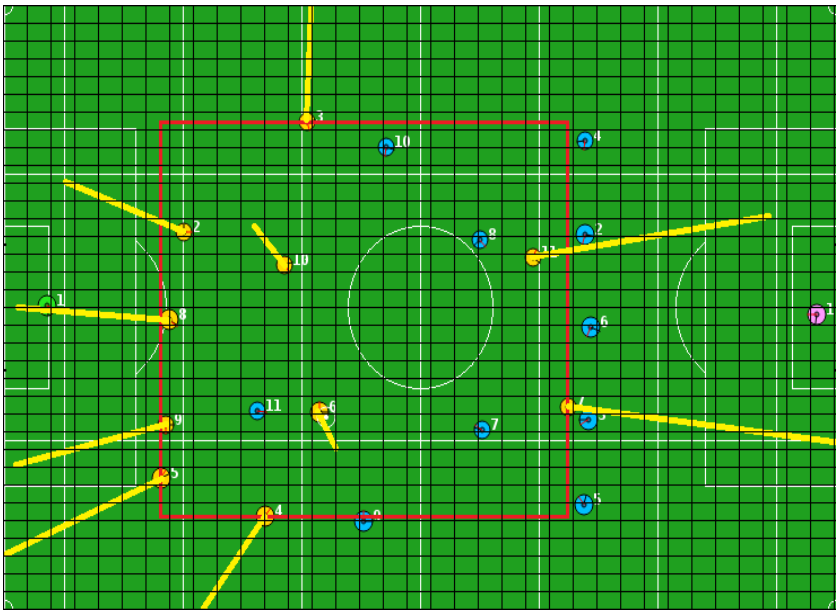


FIGURA 2.10: Mapeo de posiciones de cada jugador al campo de referencia.

Por último, se busca conocer en cada una de las matrices cuál es el área donde el jugador asociado a dicha matriz tuvo más actividad durante los ciclos transcurridos, es decir, el elemento de su matriz que tiene el número más grande. Esto se puede visualizar con mayor facilidad en la figura 2.11, donde los puntos negros indican el área del campo donde cada jugador tuvo mayor actividad durante el juego, y son estos los que proprocionarían la formación. Tomando el área de mayor actividad para cada jugador, se divide el campo, visto de forma horizontal, en 3 áreas verticales, lo que dará como resultado una formación del tipo X-Y-Z, según la cantidad de jugadores que tuvo su máxima actividad en cada una de las secciones, donde X es el número de defensores del equipo, Y el número de mediocampistas y Z el número de delanteros.

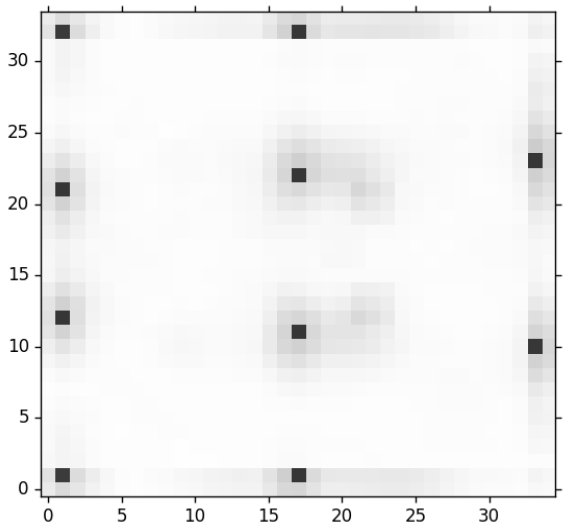


FIGURA 2.11: Matrices superpuestas de todos los jugadores.

Este proceso de reconocimiento de formaciones puede ser utilizado antes de comenzar un partido para obtener una formación promedio del equipo oponente que será almacenada y posteriormente leída por el coach al comenzar cada juego; de esta manera puede generar un planteamiento táctico inicial para el equipo JEMV, donde los datos serán extraídos de los partidos anteriores en los que haya estado presente el equipo oponente.

Además, el reconocimiento puede ser usado en tiempo real. Se irán almacenando en las matrices las posiciones relativas de los jugadores en los ciclos que transcurrieron hasta el momento en el cual el coach decide verificar nuevamente la formación oponente. La verificación y análisis de los datos para obtener la nueva formación del equipo oponente ocurre al momento de recibir algún gol del oponente después del ciclo 1.750 del partido. Se escogió este ciclo para tener información suficiente que se precisa en la identificación de la formación del oponente hasta el momento. Se realiza el proceso cuando el oponente anota un gol, ya que esto es indicativo de que el equipo JEMV no está logrando contrarrestar correctamente el esquema táctico oponente.

2.4. Estrategias para contrarrestar la formación del oponente

Habiendo obtenido la información de la formación del equipo oponente, es necesario ahora hacer uso de ésta y de otros elementos del estado del partido para definir qué formaciones usará JEMV y así lograr mejorar su desempeño durante el juego y contrarrestar al equipo oponente.

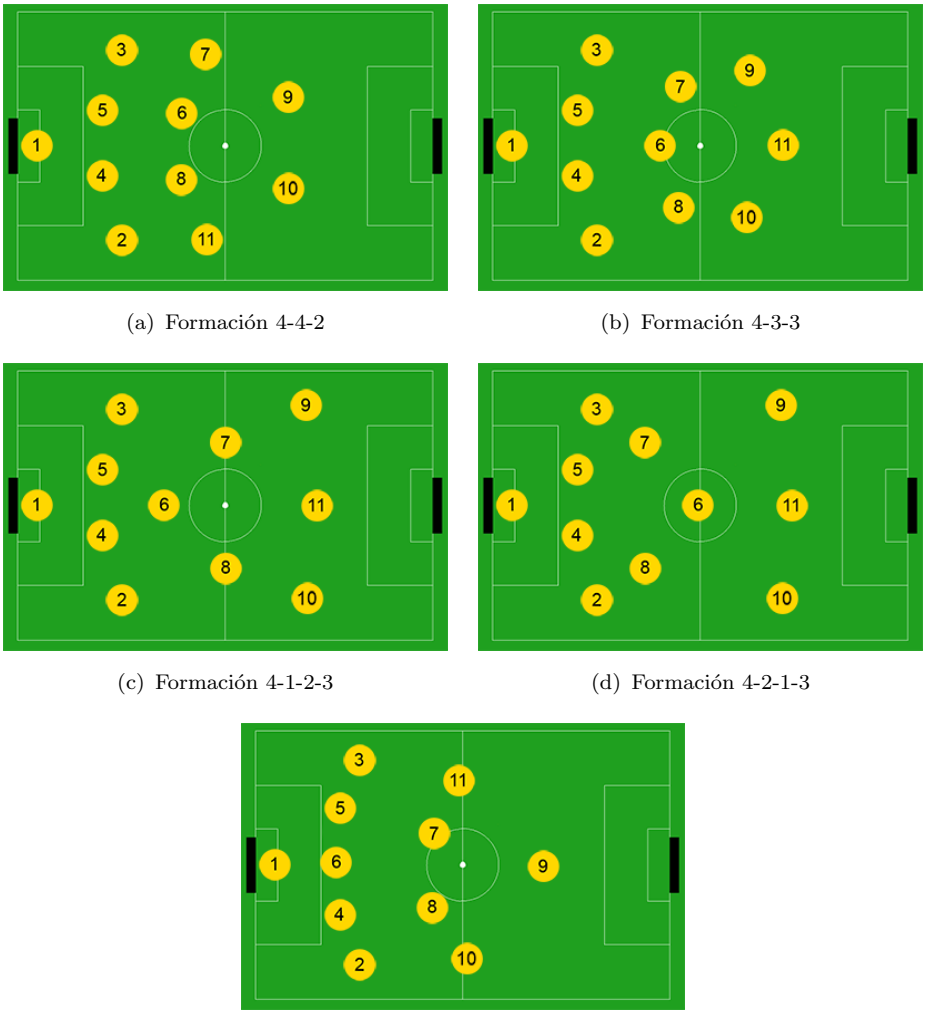
La primera condición a tomar en cuenta para definir qué formación usar es conocida como la situación del partido. El equipo *Agent2D Base* tiene por defecto la capacidad de distinguir entre tres tipos de situaciones basadas en el estado del juego: situación defensiva, situación ofensiva y situación normal. Se considera que la situación es defensiva si algún jugador del equipo oponente puede llegar al balón al menos dos ciclos de juego antes que cualquier jugador del equipo JEMV. Una situación es ofensiva cuando algún jugador del equipo JEMV puede llegar al balón al menos dos ciclos antes que cualquier jugador oponente. Por último, una situación normal es cualquiera que no entre en ninguna de las dos mencionadas anteriormente. Estas situaciones ocurren cuando el modo de juego es *PlayOn*, es decir, la pelota se encuentra en movimiento dentro del campo.

Cada formación utilizada por JEMV tiene tres modalidades para ser usadas durante el tiempo de juego del partido, asociadas a las posibles situaciones definidas anteriormente. Adicionalmente, cada formación tiene modalidades para los demás modos de juego y eventos que pueden ocurrir durante el partido como *BeforeKickOff*, para los saques al principio de cada tiempo del partido y después de un gol, *CornerKick* para cuando hay un tiro de esquina, *FreeKick* para cuando hay un tiro libre, *GoalKick* para usar cuando se realice un saque de arco y *GoalieCatch* cuando algún portero ataja el balón. En este documento se tomarán en cuenta únicamente las utilizadas durante el modo de juego *PlayOn*, ya que las otras son modalidades de casos especiales que se emplean solamente en eventos específicos.

El equipo *Agent2D Base* contiene tres formaciones básicas inicialmente, una para cada situación del partido, y alterna entre estas durante su desarrollo. Para mejorar el desempeño del equipo, se buscó crear nuevas formaciones usando el software Fedit2, implementado por los mismos creadores del equipo base. La idea inicial era crear distintas formaciones, cada una con sus modalidades de defensa,

ofensa y normal, pero, al integrar estas en el equipo, el desempeño en partidos resultó ser igual o peor que el del equipo *Agent2D Base*. Se tomó como medida la diferencia de goles entre JEMV y los equipos oponentes en una muestra de 250 partidos, 50 contra cada equipo. Por estos resultados, se decidió hacer uso de las formaciones ya existentes del equipo Genius y adaptarlas a JEMV. Las formaciones de Genius están disponibles de manera pública junto con el ejecutable del equipo.

Se hará uso de cinco formaciones del equipo Genius, cada una con sus respectivas modalidades para cada situación y evento del partido. Estas formaciones son: 4-4-2, 4-4-3, 4-1-2-3, 4-2-1-3 y 5-4-1, y se muestran en la figura 2.12.



(a) Formación 4-4-2

(b) Formación 4-3-3

(c) Formación 4-1-2-3

(d) Formación 4-2-1-3

(e) Formación 5-4-1

FIGURA 2.12: Formaciones del equipo Genius usadas por JEMV en este proyecto.

Después de haber identificado la situación del partido, se procede a realizar otras verificaciones basadas en la formación del oponente, el número de goles anotado por cada equipo y el número de ciclos transcurridos del partido. Estas verificaciones permitirán decidir qué formación usar y, según la situación, se escogerá la modalidad adecuada.

- Cuando el ciclo actual es mayor o igual a 4.200 (es el último tercio del segundo tiempo):
  - Si el oponente está ganando por menos de dos goles, utilizar 4-3-3.
  - Si el oponente está ganando por dos goles o más, utilizar 4-4-2.
  - Si el oponente está perdiendo y tiene mayor cantidad de delanteros que mediocampistas, utilizar 5-4-1.
  - Si el oponente está perdiendo y tiene menor o igual cantidad de delanteros y mediocampistas, utilizar 4-4-2.

- Cuando el ciclo del partido es menor a 4.200:
  - Si la formación del oponente es 4-3-3, se utiliza 4-1-2-3.
  - Si el oponente tiene más delanteros que mediocampistas, utilizar 4-4-2.
  - Si el oponente tiene 5 o más mediocampistas, se utiliza 5-4-1.
  - Si el oponente tiene menos de 5 mediocampistas, se utiliza 4-2-1-3.

Estas condiciones son verificadas en el orden en el que aquí aparecen, por lo tanto se emplea la formación asociada a la primera condición que se cumpla.

Las condiciones mostradas buscan que el equipo contrarreste la formación del oponente y los eventos que están ocurriendo en el partido en un momento dado. Por ejemplo, si el oponente está perdiendo al final del partido y tiene muchos delanteros (se vuelve muy ofensivo), se utiliza una formación 5-4-1 que es defensiva con el objetivo de mantener la ventaja que se tiene. Por otro lado, si el oponente está ganando por pocos goles se hace uso de una formación 4-3-3 que es un poco más ofensiva para intentar anotar goles. En el caso de que el oponente esté ganando por 2 goles o más, se adopta una formación 4-4-2 para balancear un poco el aspecto defensivo para evitar más goles en contra. La idea general de estas condiciones es

mantener un esquema defensivo la mayoría del tiempo, ya que los equipos a los que JEMV se enfrentará son mucho más complejos y hay mayores posibilidades de que anoten goles y ganen si se adopta un esquema de formación ofensivo.

De todas las posibles combinaciones de condiciones, estas fueron seleccionadas porque con ellas se obtuvieron los mejores resultados luego de realizar pruebas de 250 partidos y con diferencia de goles como métrica de desempeño. Asimismo, el resto de las condiciones con las que se realizaron pruebas obtuvieron diferencias de goles promedio mucho peores a las aquí planteadas, que se asemejaban a los resultados obtenidos con el equipo *Agent2D Base*. En el capítulo 3 se explicará con mayor detalle el proceso de experimentos realizado y los resultados obtenidos en ellos.

Adicionalmente, el equipo *Agent2D Base* realiza una asignación de tipo de jugador a los distintos agentes del equipo al comienzo del partido. Cada tipo de jugador tiene unas características que incluyen velocidad, resistencia, distancia a la que puede patear el balón y la velocidad con la cual se recupera su resistencia al realizar acciones. Estas asignaciones han sido modificadas ligeramente para el equipo JEMV. Cada tipo puede ser asignado solo una vez.

Para asignar el tipo de jugador a cada agente se toma en cuenta su eficiencia en el campo; dicha eficiencia es calculada tomando en cuenta su resistencia, velocidad y período de recuperación. Las asignaciones de tipos de jugador para JEMV y *Agent2D Base* difieren un poco, pues se modificaron las asignaciones originales con el objetivo de obtener mejores resultados.

Agent2D Base coloca a algunos jugadores ofensivos y defensivos como los más eficientes, seguidos de ciertos mediocampistas, varios jugadores ofensivos y defensivos y culminan la lista con los mediocampistas finales.

En el caso de JEMV, se tomó la decisión de que el lado defensivo y ofensivo de la formación fuesen los más eficientes, y se dejaron a los mediocampistas como los menos eficientes. La idea de esto es que como los jugadores mediocampistas no tienen la necesidad de moverse en distancias tan largas, ni tan constantemente, es válido dejarlos como los menos eficientes, mientras que los jugadores delanteros y defensas necesitan mayor eficiencia para cubrir largas distancias en menor tiempo.

Capítulo 3

Experimentos y Resultados

En este capítulo se presentan los experimentos realizados para evaluar las dos funcionalidades desarrolladas para el coach del equipo JEMV y el análisis de los resultados obtenidos.

Los experimentos se encuentran divididos en dos secciones principales. La primera sección busca evaluar la precisión de los árboles de decisión en la clasificación de acciones durante un partido. Se tomarán en cuenta modelos para cada tipo de acción (dribles, pases y tiros al arco) y para cada equipo oponente.

La segunda sección busca determinar si existe un impacto estadísticamente significativo en el equipo JEMV cuando este usa: (a) el reconocimiento de las acciones en los jugadores y en el *coach*, (b) la detección de las formaciones oponentes y el planteamiento de un esquema táctico que las contrarreste, y (c) ambas funcionalidades en conjunto.

En caso de que exista una diferencia estadística significativa entre las diferentes versiones del equipo JEMV y el equipo base, se determinará, con base en algunos valores medidos en los partidos utilizados, si las diferencias encontradas son positivas, en caso de que mejore el desempeño del equipo, o negativas, en caso de que el uso de alguna de estas características ocasionara resultados peores que los del equipo base.

3.1. Evaluación de árboles de decisión

Para evaluar la calidad de los árboles de decisión utilizados en la clasificación de acciones, se realizaron experimentos con modelos separados por tipo de acción.

Los modelos de árboles de decisión utilizados en esta sección y durante todo el proyecto fueron los provistos por la librería OpenCV en su versión 2.4 para el lenguaje C++.

En las secciones 3.1.1, 3.1.2 y 3.1.3 se mostrarán los resultados de la clasificación de dribles, pases y tiros al arco, respectivamente, separados por equipos.

Las métricas de performance empleadas para medir la calidad de los modelos fueron precisión, para evaluar la cantidad de datos clasificados correctamente, y matriz de confusión, para detectar una posible parcialización de los resultados. Además, se empleó la técnica de validación cruzada *K-fold* tomando *K=5* para la fase de entrenamiento.

3.1.1. Clasificación de dribles

El objetivo de este experimento es evaluar el desempeño de los árboles de decisión para la clasificación de dribles. El modelo fue entrenado únicamente con acciones de dribles exitosos y dribles insatisfactorios del equipo *Agent2D Base*.

De los 350 partidos entre el equipo *Agent2D Base* y Genius se extrajeron 14.620 datos que fueron utilizados en la fase de entrenamiento, de los cuales 7.310 son datos clasificados como 'Dribles', y 7.310 clasificados como 'Dribles fallidos'. Para la fase de pruebas se utilizaron 100 partidos entre ambos equipos; se obtuvieron así 3.270 datos, divididos equitativamente entre cada clase.

Los resultados del porcentaje de aciertos en cada fase se encuentran en la Tabla 3.1. Se observa que el modelo en la fase de entrenamiento clasificó correctamente 10.756 datos, que representan el 73,5704 % del total. En la fase de prueba, donde se utilizaron datos nuevos, el modelo disminuyó su exactitud pues clasificó correctamente 2.307 de los datos, que representan el 70,5505 % del total.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	14.620	73,5704	26,4295
Prueba	3.270	70,5505	29,4495

Tabla 3.1: Exactitud del árbol de decisión para clasificación de dribles contra Genius en fases de entrenamiento y prueba

Además de los porcentajes de aciertos antes presentados para Genius, en la Tabla 3.2 se tiene la matriz de confusión para la fase de entrenamiento. El modelo para este equipo no parece estar desbalanceado hacia algunas de las clases, puesto que los porcentajes de falsos positivos (dribles fallidos clasificados como exitosos) y falsos negativos (dribles exitosos clasificados como fallidos) se encuentran entre 26 y 28 %.

K=5	Clasificados 'Drible'	Clasificados 'Drible fallido'
'Drible'	5.314	1.996
'Drible fallido'	1.868	5.442

Tabla 3.2: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de dribles contra Genius

De los 350 enfrentamientos entre *Agent2D Base* y el equipo Helios se obtuvieron 14.690 dribles, de los cuales 7.345 fueron etiquetados como exitosos y 7.345 como fallidos. Para la fase de pruebas se realizaron 100 partidos entre ambos equipos, lo que generó 700 datos.

En la Tabla 3.3 se encuentran los porcentajes de aciertos tanto para la fase de entrenamiento como para la de pruebas. En el entrenamiento se clasificaron correctamente 10.796 datos, representando el 73,4922 % del total, y en la fase de pruebas se realizó una predicción correcta en 654 ocasiones, es decir, el 93,4286 % de los datos de prueba.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	14.690	73,4922	26,5078
Prueba	700	93,4286	6,57143

Tabla 3.3: Exactitud del árbol de decisión para clasificación de dribles contra Helios en fases de entrenamiento y prueba

Al igual que con el equipo anterior, la predicción para Helios no parece estar parcializada hacia alguna de las dos clases, como se puede comprobar en la Tabla 3.4. Clasificó correctamente el 75,22 % de los dribles exitosos y 71,76 % de los dribles fallidos.

K=5	Clasificados 'Drible'	Clasificados 'Drible fallido'
'Drible'	5.525	1.820
'Drible fallido'	2.074	5.271

Tabla 3.4: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de dribles contra Helios

El modelo contra el equipo Hermes fue entrenado y probado con 9.310 y 1.330 datos, respectivamente. De los resultados presentados en la Tabla 3.5 se puede deducir que el modelo de dribles para este equipo tiene mayores dificultades para generalizar que para los dos equipos anteriores, esto a causa de que al presentarle datos nuevos en la fase de prueba, disminuyó considerablemente el porcentaje de aciertos.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	9.310	78,174	21,826
Prueba	1.330	68,4211	31,5789

Tabla 3.5: Exactitud del árbol de decisión para clasificación de dribles contra Hermes en fases de entrenamiento y prueba

A pesar de presentar una disminución de la exactitud en la fase de prueba, este hecho no parece ser causado por una parcialización del modelo durante el entrenamiento. Esto puede ser observado en la Tabla 3.6 que contiene la matriz de confusión para la fase de entrenamientos. El modelo clasificó correctamente 120 datos más para los dribles exitosos que para los fallidos, lo que representa solo un 3 % de diferencia entre una clase y otra.

K=5	Clasificados 'Drible'	Clasificados 'Drible fallido'
'Drible'	3.699	956
'Drible fallido'	1.076	3.579

Tabla 3.6: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de dribles contra Hermes

Para el equipo Jaeger se generaron 13.400 datos con el objeto de entrenar el modelo y 1.986 para evaluar su generalización. Los resultados obtenidos al evaluar el desempeño en ambos procesos son similares a los del equipo Helios y se encuentran en la Tabla 3.7. Se obtuvo un porcentaje de aciertos mayor en la fase de entrenamiento, pues clasificó correctamente el 77,3881 % de los datos, pero este porcentaje disminuyó hasta el 65,1057 % al probar el modelo con nuevos datos.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	13.400	77,3881	22,6119
Prueba	1.986	65,1057	34,8943

Tabla 3.7: Exactitud del árbol de decisión para clasificación de dribles contra Jaeger en fases de entrenamiento y prueba

En la Tabla 3.8 se tiene con mayor detalle los resultados obtenidos durante el entrenamiento del modelo para el equipo Jaeger. El modelo no parece estar sesgado, puesto que solo se obtuvo aproximadamente un 2 % más de falsos positivos que negativos.

K=5	Clasificados 'Drible'	Clasificados 'Drible fallido'
'Drible'	5.265	1.435
'Drible fallido'	1.595	5.105

Tabla 3.8: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de dribles contra Jaeger

Finalmente, para el equipo Wright Eagle se tienen 16.230 datos que servirán como entrenamiento para el modelo y 1.762 datos para probarlo. Los resultados de la precisión de los árboles en ambas fases se encuentra en la Tabla 3.9, donde se puede observar que es el equipo, de los cinco utilizados, que tiene mayores problemas para clasificar correctamente los dribles, pues solamente lo hace con un porcentaje de 60,1022 de exactitud en la fase de pruebas.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	16.230	77,3629	22,6371
Prueba	1.762	60,1022	39,8978

Tabla 3.9: Exactitud del árbol de decisión para clasificación de dribles contra Wright Eagle en fases de entrenamiento y prueba

Al ver la matriz de confusión obtenida del entrenamiento del modelo presente en la Tabla 3.10, se puede comprobar que la dificultad para la clasificación de datos nuevos no parece estar asociada a una parcialización.

K=5	Clasificados 'Drible'	Clasificados 'Drible fallido'
'Drible'	6.374	1.741
'Drible fallido'	1.933	6.182

Tabla 3.10: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de dribles contra Wright Eagle

3.1.2. Clasificación de pases

Los experimentos de esta sección fueron realizados para evaluar el desempeño de árboles de decisión en la clasificación de pases, tanto satisfactorios como fallidos, para cada uno de los cinco equipos oponentes seleccionados.

Para entrenar el árbol de decisión contra el equipo Genius se utilizaron 10.410 datos, divididos con la misma proporción para datos de las clases 'Pase' y 'Pase fallido'. Estos datos son producto de la generación de 350 partidos entre el equipo *Agent2D Base* y Genius. En la fase de pruebas se emplearon 2.354 datos extraídos de 100 partidos nuevos entre ambos equipos.

Los resultados del porcentaje de aciertos producto de la clasificación de los datos en ambas clases se tienen en la Tabla 3.11. La proporción de datos bien clasificados se mantiene al poner el modelo a prueba con datos no vistos en la fase de entrenamiento. Esto hace pensar que el modelo de pases para Genius no tiene problemas de generalización con datos nuevos, como ocurrió con algunos modelos de la sección anterior.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	10.410	71,9597	28,0403
Prueba	2.354	73,1946	26,8054

Tabla 3.11: Exactitud del árbol de decisión para clasificación de pases contra Genius en fases de entrenamiento y prueba

Además, la cantidad de datos clasificados de forma correcta por clase se encuentra casi totalmente equilibrada, es decir, que el modelo no está parcializado. Estos datos están presentes en la Tabla 3.12. Allí se puede notar que solo se clasificaron correctamente 25 datos más de 'Pase' que de 'Pase fallido'.

K=5	Clasificados 'Pase'	Clasificados 'Pase fallido'
'Pase'	3.758	1.447
'Pase fallido'	1.472	3.733

Tabla 3.12: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de pases contra Genius

De los enfrentamientos entre el equipo base y Helios se extrajeron 10.410 acciones, de las que fueron clasificadas correctamente 7.585, lo que representa el 72,8626 % de los datos de entrenamiento, como se muestra en la Tabla 3.13. En la fase de pruebas se obtuvo un resultado similar, pues se clasificaron de forma correcta 404 del total de datos, representando el 73,4545 %.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	10.410	72,8626	27,1374
Prueba	550	73,4545	26,5455

Tabla 3.13: Exactitud del árbol de decisión para clasificación de pases contra Helios en fases de entrenamiento y prueba

Además, en la Tabla 3.14 se tiene la matriz de confusión correspondiente a la fase de entrenamiento para el equipo Helios. En ella se observa que el modelo parece estar generalizando de la misma forma la predicción entre ambas clases.

K=5	Clasificados 'Pase'	Clasificados 'Pase fallido'
'Pase'	3.766	1.439
'Pase fallido'	1.386	3.819

Tabla 3.14: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de pases contra Helios

Los resultados de la exactitud contra el equipo Hermes se encuentran en la Tabla 3.15. Se puede comprobar que de los 8.370 datos utilizados para entrenar el

modelo se clasificaron de forma correcta 6.096, lo que representa el 72,8315 %. En la fase de pruebas se emplearon 1.170 datos, y el modelo clasificó correctamente 850 de ellos.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	8.370	72,8315	27,1685
Prueba	1.170	72,6496	27,3504

Tabla 3.15: Exactitud del árbol de decisión para clasificación de pases contra Hermes en fases de entrenamiento y prueba

A partir de los datos presentes en la Tabla 3.16 se puede concluir que el modelo para Hermes está teniendo las mismas dificultades con los datos de ambas clases, pues genera 26,24 % de falsos positivos y 28,10 % de falsos negativos.

K=5	Clasificados 'Pase'	Clasificados 'Pase fallido'
'Pase'	3.009	1.176
'Pase fallido'	1.098	3.087

Tabla 3.16: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de pases contra Hermes

Los resultados para el equipo Jaeger se encuentran en la Tabla 3.17. En la etapa de experimentos se utilizaron 9.640 datos y se clasificaron correctamente 6.907, lo que representa el 71,6494 %. Además, durante la fase de prueba del modelo se emplearon 1.386 datos nuevos, de los cuales se clasificaron de forma correcta 1.012, lo que es el 73,0159 % del total.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	9.640	71,6494	28,3506
Prueba	1.386	73,0159	26,9841

Tabla 3.17: Exactitud del árbol de decisión para clasificación de pases contra Jaeger en fases de entrenamiento y prueba

El modelo de pases contra Jaeger parece estar bastante equilibrado en cuanto a la clasificación de las dos modalidades de la acción, como se puede observar en

los datos del cuadro 3.18. Allí se aprecia que las proporciones de falsos positivos y falsos negativos son similares, es decir, de 28,44 % y 28,25 %, respectivamente.

K=5	Clasificados 'Pase'	Clasificados 'Pase fallido'
'Pase'	3.458	1.362
'Pase fallido'	1.371	3.449

Tabla 3.18: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de pases contra Jaeger

Por último, se presentan los resultados del modelo de pases para el equipo Wright Eagle en la Tabla 3.19. En el entrenamiento fueron utilizados 16.390 datos, divididos de forma equitativa entre ambas clases, y de los cuales se clasificaron exitosamente 11.429, lo que supone el 69,7315 % del total. A su vez, para probar la generalización del modelo se emplearon 2.398 datos nuevos, de los cuales 1.716 se clasificaron correctamente.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	16.390	69,7315	30,2685
Prueba	2.398	71,5596	28,4404

Tabla 3.19: Exactitud del árbol de decisión para clasificación de pases contra WrightEagle en fases de entrenamiento y prueba

Al igual que el resto de los modelos de pases para los otros equipos, el del equipo Wright Eagle no parece encontrarse parcializado. Los datos presentes en la Tabla 3.20 indican que la cantidad de falsos positivos y falsos negativos es aproximadamente la misma, lo que representa el 30,14 % y 30,39 % de los datos de cada clase.

K=5	Clasificados 'Pase'	Clasificados 'Pase fallido'
'Pase'	5.704	2.491
'Pase fallido'	2.470	5.725

Tabla 3.20: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de pases contra Wright Eagle

3.1.3. Clasificación de tiros al arco

En esta sección se presentarán los experimentos para la clasificación de los tiros al arco tanto exitosos como fallidos usando árboles de decisión.

El árbol de decisión para la clasificación de tiros al arco contra el equipo Genius fue entrenado con 740 datos, de los cuales el 73,3784 % se clasificó correctamente. Las pruebas para evaluar la generalización con datos nuevos se realizaron con 272 datos; de esta manera se logró predecir de forma exitosa la clase del 75,7353 % de los datos.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	740	73,3784	26,6216
Prueba	272	75,7353	24,2647

Tabla 3.21: Exactitud del árbol de decisión para clasificación de tiros al arco contra Genius en fases de entrenamiento y prueba

Este modelo de árboles presentó más problemas para clasificar datos de la clase 'Tiro al arco' que de la clase 'Tiro al arco fallido'. De los datos presentes en la Tabla 3.22 se tiene que el porcentaje de falsos negativos es de 30,81 % y el de falsos positivos de 23,49 %.

K=5	Clasificados 'Tiro al arco'	Clasificados 'Tiro al arco fallido'
'Tiro al arco'	256	114
'Tiro al arco fallido'	83	287

Tabla 3.22: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de tiros al arco contra Genius

La cantidad de tiros al arco, tanto exitosos como fallidos, producto de los partidos contra el equipo Helios, es considerablemente menor que la del resto de los equipos. Esto se debe a la gran dificultad que tiene el equipo *Agent2D Base* para anotarle goles. A pesar de esto, se lograron extraer 220 datos, igualmente divididos entre ambas clases para la fase de entrenamiento, y 66 datos para la fase de pruebas.

Los resultados que aparecen en la Tabla 3.23 muestran que aunque la cantidad de datos utilizada para entrenar no es tan grande como para los otros modelos, puede que no esté afectando a la generalización.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	220	74,5455	25,4545
Prueba	66	74,2424	25,7576

Tabla 3.23: Exactitud del árbol de decisión para clasificación de tiros al arco contra Helios en fases de entrenamiento y prueba

De la matriz de confusión generada con los resultados de la fase de entrenamiento, presente en la Tabla 3.24, se tiene que el modelo tuvo la misma dificultad en la predicción de datos para ambas clases.

K=5	Clasificados 'Tiro al arco'	Clasificados 'Tiro al arco fallido'
'Tiro al arco'	82	28
'Tiro al arco fallido'	28	82

Tabla 3.24: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de tiros al arco contra Helios

Para el equipo Hermes se extrajeron 420 datos que fueron empleados en la etapa de entrenamiento, de los cuales se clasificaron correctamente 297, lo que representa el 70,7143 % del total. En la etapa de prueba del modelo se utilizaron 208 datos, y se clasificó correctamente el 80,3828 %.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	420	70,7143	29,2857
Prueba	208	80,3828	19,6172

Tabla 3.25: Exactitud del árbol de decisión para clasificación de tiros al arco contra Hermes en fases de entrenamiento y prueba

La cantidad de datos clasificados exitosamente para ambas clases con el modelo de Hermes es similar, como puede observarse en la Tabla 3.26. Con los datos obtenidos se tiene que el modelo arrojó un 27,62 % de falsos positivos y un 30,98 % de falsos negativos.

K=5	Clasificados 'Tiro al arco'	Clasificados 'Tiro al arco fallido'
'Tiro al arco'	145	65
'Tiro al arco fallido'	58	152

Tabla 3.26: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de tiros al arco contra Hermes

Los resultados del porcentaje de aciertos para el equipo Jaeger se encuentran en la Tabla 3.27. De los 1.200 datos extraídos de los partidos para la fase de entrenamiento, se clasificaron correctamente 905, lo que representa el 75,4167 % del total. En la fase de prueba se extrajeron 552 datos, de los cuales 423 se clasificaron correctamente.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	1.200	75,4167	24,5833
Prueba	552	76,6304	23,3696

Tabla 3.27: Exactitud del árbol de decisión para clasificación de tiros al arco contra Jaeger en fases de entrenamiento y prueba

Además, en la Tabla 3.28 se encuentra la matriz de confusión del modelo para la fase de entrenamiento, de la que se puede concluir que el modelo de tiros al arco contra el equipo Hermes parece no estar parcializado hacia ninguna de las clases.

K=5	Clasificados 'Tiro al arco'	Clasificados 'Tiro al arco fallido'
'Tiro al arco'	439	161
'Tiro al arco fallido'	134	466

Tabla 3.28: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de tiros al arco contra Jaeger

Por último, se realizaron los experimentos con el equipo Wright Eagle. En la etapa de entrenamiento fueron utilizados 430 datos, equitativamente distribuidos entre ambas clases, y para la etapa de prueba se usaron 208 datos. En la primera etapa se clasificó correctamente el 74,186 % de los datos, mientras que en la etapa de prueba fue el 85,5769 %.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	430	74,186	25,814
Prueba	208	85,5769	14,4231

Tabla 3.29: Exactitud del árbol de decisión para clasificación de tiros al arco contra Wright Eagle en fases de entrenamiento y prueba

En la Tabla 3.30 se encuentra la matriz de confusión que se obtuvo a partir de los resultados de la fase de entrenamiento del modelo con Wright Eagle. Se puede observar que con los datos obtenidos, el modelo no está parcializado, ya que solo hay una diferencia de un dato entre los falsos positivos y negativos.

K=5	Clasificados 'Tiro al arco'	Clasificados 'Tiro al arco fallido'
'Tiro al arco'	159	55
'Tiro al arco fallido'	56	160

Tabla 3.30: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de tiros al arco contra Wright Eagle

3.2. Resultados de las pruebas estadísticas finales

En esta sección se mostrarán los resultados de las pruebas estadísticas que se realizaron con el objetivo de descubrir si las funcionalidades agregadas al equipo *Agent2D Base* para transformarlo en JEMV han mejorado su desempeño en partidos.

La primera subsección estará dedicada a los resultados de las pruebas estadísticas que comparan *Agent2D Base* con el equipo JEMV; se emplea únicamente la funcionalidad de detección de acciones por medio de árboles de decisión. En la segunda subsección se encuentran los resultados de las pruebas estadísticas entre *Agent2D Base* y el equipo JEMV; se hace uso exclusivamente del reconocimiento de formaciones y estrategias. Por último, se mostrarán los resultados finales comparando el desempeño de *Agent2D Base* con el de JEMV y para ello se usarán las funcionalidades de las dos subsecciones anteriores.

Las pruebas realizadas toman como grupo de control al equipo *Agent2D Base* y como grupo experimental a JEMV con las funcionalidades asociadas a cada subsección. Para los resultados se tomaron en cuenta 50 partidos de control contra cada uno de los equipos oponentes (Genius, Helios, Hermes, Jaeger y Wright Eagle) para un total de 250 partidos, y 50 experimentales contra los mismos equipos para tener 250 partidos experimentales. Es importante destacar que los 250 partidos habidos como grupo de control son los mismos en todas las pruebas realizadas.

La métrica utilizada para determinar si hubo una mejora en el desempeño del equipo al hacer uso de las nuevas funcionalidades será la diferencia de goles de cada partido, y esta será la resta entre los goles anotados por el equipo de control o experimental y los goles anotados por el equipo oponente. Se considerará que el equipo ha mejorado su desempeño cuando la media de la diferencia de goles para el grupo experimental sea menor a la del grupo de control.

Para comparar el desempeño de ambos grupos y determinar si existe una diferencia estadísticamente significativa entre ellos, se aplicarán pruebas de tipo t de Student, y se tomará un valor para α de 0,01. Debido a que el tamaño de cada muestra es suficientemente grande, por el teorema del Límite Central se puede decir que ambos conjuntos de datos tienen una distribución aproximadamente normal. A partir de este punto, el uso de las abreviaciones M y SD servirán para referenciar la Media y Desviación Estándar respectivamente.

Además de presentar los resultados de las pruebas aplicadas, se incluirá el total de goles anotados y recibidos para cada grupo, con el objetivo de tener una visión más detallada de los experimentos.

3.2.1. Pruebas estadísticas para el reconocimiento de acciones

En esta sección se presentarán los resultados obtenidos al aplicar la prueba t de Student para evaluar la existencia de una diferencia estadísticamente significativa entre el equipo de control *Agent2D Base* y el equipo experimental JEMV, modificado únicamente para que sus jugadores y *coach* utilicen la funcionalidad de reconocimiento de acciones durante los partidos.

En la Tabla 3.31 se muestran los resultados de la prueba t de Student aplicada sobre el grupo de control y el experimental. Para esta prueba se tomó como hipótesis nula que la diferencia de goles entre ambos grupos no es estadísticamente significativa y, como hipótesis alternativa, que existe estadísticamente una diferencia significativa entre las medias.

Grupo	Media de diferencia de goles	Diferencia total de goles	Goles a favor	Goles en contra	p-valor
Control	-4,816	-1.204	222	1.426	9,857e-10
Experimental	-6,94	-1.735	24	1.759	

Tabla 3.31: Resultados de la prueba t de Student para la evaluación del reconocimiento de accciones en el equipo JEMV

A partir de los resultados obtenidos se comprueba que existe un aumento estadísticamente significativo entre las medias de las diferencias de goles del grupo de control (M=-4,81,SD=3,70), y el experimental (M=-6,94,SD=3,90); t(496,62)=6,23, p=9,857e-10. En estas condiciones se acepta la hipótesis alternativa y se consideran los resultados como un deterioro estadísticamente significativo del desempeño del equipo.

Además de los valores obtenidos para las diferencias de goles de ambos grupos, se puede observar que ocurrió un aumento de 531 goles, lo que representa una desmejora en el desempeño del equipo. Viendo más en detalle, los resultados sugieren que lo alcanzado por el equipo experimental se debe a una disminución de los goles anotados por sus jugadores (198 menos) y un aumento en la cantidad de goles anotados por los equipos oponentes (333 más).

El deterioro en el desempeño del equipo al hacer uso de esta funcionalidad puede haber ocurrido por varias razones. Una posibilidad es que esté siendo causado por las predicciones de los árboles. En ese caso, mejorar la calidad del modelo generará un impacto positivo en el desempeño del equipo. Esta es una causa válida, pero no la única posible.

El uso de los árboles en el equipo podría modificarse en busca de mejorar los resultados. Actualmente los árboles sirven para filtrar acciones ya existentes que se ejecutarían automáticamente si no existieran los árboles. Una opción sería modificar este funcionamiento para que el filtro ocurra al crear la acción en vez de justo antes de ejecutarla, y en caso de que se rechace, crear otra acción similar como alternativa. Actualmente, el jugador ejecuta la acción de quedarse con el balón cuando el modelo rechaza todas las acciones anteriores, lo cual puede estar afectando al equipo si esta acción se repite durante muchos ciclos, porque el jugador quedaría en una posición estática.

Adicionalmente, al observar al equipo jugar con esta funcionalidad implementada, se notó que los jugadores tienden a quedarse más en el área central del campo, lo cual reduce las posibilidades de anotar gol. Esto puede estar siendo ocasionado por el rechazo constante de las acciones por parte de los árboles, lo que evita que los jugadores pasen el balón o driblen en dirección a la portería oponente.

3.2.2. Pruebas estadísticas para reconocimiento de formaciones y estrategias

Los resultados mostrados en esta subsección comparan el desempeño del equipo *Agent2D Base* con el de JEMV, para ellos se emplea la funcionalidad de reconocimiento de formaciones y estrategias.

El cuadro 3.32 señala los resultados de la prueba t de Student aplicada sobre el grupo de control y experimental; para esta prueba tomamos como hipótesis nula que la diferencia entre las medias de la diferencia de goles de ambos conjuntos no es estadísticamente significativa y, como hipótesis alternativa, que la diferencia de las medias sí es estadísticamente significativa.

Grupo	Media de diferencia de goles	Diferencia total de goles	Goles a favor	Goles en contra	p-valor
Control	-4,816	-1.204	222	1.426	<2,2e-16
Experimental	-2,372	-593	203	796	

Tabla 3.32: Resultados de la prueba t de Student para la evaluación del reconocimiento de formaciones y estrategias en el equipo JEMV

Se puede constatar que hay una reducción estadísticamente significativa entre las medias de las diferencias de goles del grupo de control (M=-4,81,SD=3,70) y el experimental (M=-2,37,SD=2,53); t(439,51)=-8,60, p<2,2e16. Con esto se acepta la hipótesis alternativa y, viendo los valores de los resultados se puede afirmar que representa una mejora estadísticamente significativa del desempeño del equipo.

En la tabla también se puede ver que la diferencia de goles total para el grupo de control es de -1.204 y para el experimental es de -593, lo cual representa una reducción de 611 goles, una mejora de casi la mitad. Adicionalmente se muestra que esta mejora proviene de una defensa más eficiente en el equipo. Los goles totales en contra se reducen en 630, pero también se reducen los goles a favor en 19. La reducción en el número de goles en contra es suficientemente grande como para considerar esto una mejora en general del equipo, independientemente de que los goles a favor también se hayan reducido en una pequeña cantidad.

La mejora en los resultados de JEMV respecto al equipo base se debe a una combinación entre el uso de formaciones nuevas y más numerosas que las 3 formaciones iniciales del equipo base, y el buen uso de éstas a lo largo del partido. Las nuevas formaciones permiten al equipo cambiar su posicionamiento y estilo de juego durante el partido según la situación del mismo. El buen uso de ellas permite contrarrestar de manera efectiva las acciones de los oponentes, adoptando, por ejemplo, formaciones más defensivas si JEMV está ganando el partido, o formaciones más ofensivas o balanceadas para contrarrestar el número de oponentes en cada línea de juego (defensas, mediocampistas y delanteros).

3.2.3. Pruebas estadísticas para reconocimiento de acciones y de formaciones

Los experimentos realizados en esta subsección comparan el desempeño del equipo *Agent2D Base* con el de JEMV haciendo uso de las funcionalidades mostradas en las dos subsecciones anteriores, la detección de acciones y el reconocimiento de formaciones y elaboración de estrategias para contrarrestar al equipo oponente.

La tabla 3.33 muestra los resultados de la prueba t de Student aplicada sobre el grupo de control y el experimental. En esta prueba se tomó como hipótesis nula que la desigualdad entre las medias de la diferencia de goles de ambos conjuntos no es estadísticamente significativa y, como hipótesis alternativa, que sí existe desigualdad estadísticamente significativa entre las medias.

Grupo	Media de diferencia de goles	Diferencia total de goles	Goles a favor	Goles en contra	p-valor
Control	-4,816	-1.204	222	1.426	0,0019
Experimental	-3,94	-985	25	1.010	

Tabla 3.33: Resultados de la prueba t de Student para la evaluación del reconocimiento de acciones y de formaciones en conjunto en el equipo JEMV

Como se puede observar, existe una reducción estadísticamente significativa en las medias de las diferencias de goles del grupo de control (M=-4,81,SD=3,70) y el experimental (M=-3,94,SD=2,45); t(431,82)=-3,11, p=0,0019. Con estos resultados podemos aceptar la hipótesis alternativa y decir que el uso de las dos funcionalidades en el equipo representa una mejora estadísticamente significativa para el mismo.

La tabla también nos muestra que la diferencia de goles total para el grupo de control es de -1.204 y para el experimental es de -985, lo cual representa una reducción de 219 goles. Este resultado significa una mejora en el desempeño del equipo respecto al grupo de control, pero un deterioro comparado con los resultados en la subsección de pruebas asociadas a la detección de formaciones y estretegias. En la tabla se puede apreciar también que hubo una reducción de los goles a favor

del equipo, pues pasaron de 222 en el grupo de control a 25 en el experimental. Se tuvo una reducción de 197 goles a favor, pero también hubo una reducción en los goles en contra de 416 goles menos en el grupo experimental respecto del grupo de control.

Los resultados al combinar las dos funcionalidades anteriores eran de esperarse, ya que se está combinando el deterioro en el desempeño que se mostró al incluir los árboles de decisión y sus predicciones en el equipo, con la mejora obtenida al aplicar la detección de acciones y planteamiento de estrategias.

Se buscó combinar ambas funcionalidades con el objetivo de observar si la mejora en el posicionamiento del equipo con el uso de formaciones nuevas y métodos para contrarrestar al oponente se complementaban con la detección y predicción de acciones y mostrar resultados aún mejores en el equipo, pero este no fue el caso.

Capítulo 4

Conclusiones y Recomendaciones

4.1. Conclusiones

En el proyecto realizado se logró implementar un *coach* para el equipo *Agent2D Base* que, por medio del uso de *logs* de partidos anteriores y la información obtenida en tiempo real, mejora el desempeño del equipo JEMV de manera considerable contra equipos participantes de competencias de Robocup pasadas.

Una de las funcionalidades implementadas en el *coach* de JEMV le permite hacer uso de *logs* de partidos pasados para extraer información sobre las acciones realizadas, y con ella generar modelos de árboles de decisión que posibilitan distinguir entre acciones exitosas y fallidas en tiempo real.

Las pruebas efectuadas en los árboles de decisión arrojaron resultados sobre el 70 % de precisión en la mayor parte de los modelos, resultados separados por tipos de acciones y equipos, con la excepción de los modelos de dribles de los equipos Wright Eagle y Jaeger, en estos se alcanzó una precisión de 60 % y 65 %, respectivamente. Estos resultados demuestran que es posible construir modelos de aprendizaje de máquina que permitan aprovechar datos de partidos pasados con la finalidad de convertirlos en información para mejorar la toma de decisiones de los jugadores.

Con la finalidad de mejorar los resultados generados por los árboles de decisión,

se implementaron redes neuronales para la clasificación de acciones. Los resultados obtenidos por este modelo superan también el 70 % de aciertos, incluso llegando a un 80 % en algunos casos. A pesar de esto, se decidió seguir usando los árboles de decisión porque los resultados de las redes no presentaron gran diferencia respecto a los de los árboles, y el tiempo de entrenamiento de las redes dificultaba su uso en tiempo real, factor determinante para que el *coach* pueda realizar su trabajo durante los partidos.

Los porcentajes de aciertos generados tanto por los árboles de decisión como por las redes neuronales para las fases de entrenamiento lleva a pensar que el problema de la clasificación no reside en el tipo de algoritmo de aprendizaje de máquina utilizado. Se cree entonces que el problema radica en la naturaleza de los datos. Es posible que la precisión de la clasificación de estos modelos pueda incrementarse si se realiza un estudio más detallado de los atributos utilizados para modelar el problema, así como de las relaciones entre ellos. Tomar en cuenta información de los partidos distinta a la presentada en el Capítulo 3 podría generar atributos más significativos para los modelos.

Al hacer uso de los árboles de decisión en partidos reales, el desempeño del equipo JEMV se ve significativamente reducido en comparación con el de *Agent2D Base*. Esto indica que no basta con que los árboles de decisión puedan clasificar correctamente las acciones, es necesario también que los jugadores del equipo tengan alternativas o estrategias adicionales para aplicar en los casos en los que las predicciones de los árboles indiquen que la acción a realizar no será exitosa.

Otra de las posibles causas del deterioro del desempeño del equipo al usar esta funcionalidad es la manera en la que se están empleando los árboles de decisión. Actualmente estos modelos sirven como filtro para que cada jugador al momento de tener la pelota decida si ejecuta una acción o no. Esta toma de decisiones ocurre luego que el jugador ya construyó su cadena de acciones posibles, y en caso de que los árboles las descarten todas, el jugador retiene el balón. Realizar la predicción antes de que el jugador cree su cadena de acciones posibles podría permitir generar nuevas alternativas en caso de rechazo.

Por otro lado, se logró implementar en el *coach* una funcionalidad que le permite hacer uso de la información obtenida a partir de *logs* de partidos anteriores, así como en tiempo real, para determinar la formación promedio usada por el equipo

oponente. El método usado para esto se basó en el trabajo de (Habibi et al., 2002), pero en lugar de retornar información general sobre el equipo oponente y sus movimientos, hace uso de una matriz dedicada a cada jugador, lo cual permite obtener información específica de dicho jugador e identificar con mayor claridad la formación del equipo completo.

Teniendo la formación del oponente y haciendo uso de otra información del estado del partido actual, como el ciclo del partido y la cantidad de goles anotada por cada equipo, se desarrollaron estrategias que contrarrestan el esquema táctico del equipo oponente por medio de condiciones que, al cumplirse, generan cambios en la formación de JEMV. Esto permite que el equipo se adapte a su oponente y a la situación en la que se encuentra. Los resultados de partidos reales al aplicar esta funcionalidad en JEMV muestran una mejora considerable en el desempeño del equipo, reduciendo la diferencia total de goles en los partidos de prueba en más 50 % con respecto a la diferencia total de goles de *Agent2D Base*.

Al combinar las dos funcionalidades antes mencionadas, los resultados del equipo JEMV continúan generando una mejora estadísticamente significativa sobre el desempeño de *Agent2D Base*. Sin embargo, estos resultados no son mejores que cuando se utilizó únicamente la funcionalidad de detección de formaciones y planteamiento de estrategias. Esto se debe a que los resultados en tiempo real del equipo, que hacen uso de los árboles de decisión, no son buenos.

Es importante destacar que ninguna de las funcionalidades agregadas al equipo utiliza los mensajes que tiene permitido enviarle el *coach* a sus jugadores a través del servidor, puesto que simplemente se actualizaron archivos de texto para las formaciones y modelos para el caso de la detección de acciones. Esto es un factor positivo si se piensa que agregar nuevas funcionalidades que sí necesiten de estos mensajes en un futuro, no causará conflictos con las limitaciones que impone el servidor para la comunicación.

Observando de forma general los logros alcanzados y los resultados que se obtuvieron con cada una de las funcionalidades, se considera que los objetivos planteados al inicio de esta investigación fueron cumplidos con éxito. Sin embargo, el camino transitado hasta la última versión del equipo JEMV se vio plagado de las dificultades que viven todos aquellos que intentan dar sus primeros pasos dentro del ambiente de RoboCup, en específico en las categorías de simulación.

La falta de documentación oficial sobre el equipo *Agent2D Base* y sobre el ambiente de RoboCup en general, ralentizó de forma significativa el proceso de desarrollo de esta investigación. Por esto, una parte considerable del tiempo de elaboración de este proyecto fue invertido en comprender el funcionamiento del equipo, las características tanto de los jugadores como del *coach* que ya se encontraban implementadas y cómo se realizaba la interacción entre éstos y el servidor del juego.

Otra de las dificultades encontradas durante el transcurso del proyecto fue la imposibilidad de replicar los resultados del trabajo utilizado como base para la detección de acciones durante los partidos. Aun cuando se siguieron todas las explicaciones expuestas en el trabajo de Karimi y Ahmazadeh (2014), y se realizó la selección y procesamiento de los atributos de los árboles de decisión como allí se planteaba, nunca se lograron obtener porcentajes de precisión tan altos como los que dicen haber obtenidos los autores con sus modelos. Esto se tradujo en una inversión adicional de tiempo para reconsiderar y mejorar cada uno de los procesos que se estaban llevando a cabo, desde la extracción de las acciones, hasta la selección del modelo de aprendizaje de máquina empleado.

Por último, es importante destacar que los equipos a los que se enfrentó JEMV, y contra los cuales se logró mejorar significativamente el desempeño de sus jugadores durante los partidos, han sido desarrollados por grupos de investigación durante años, en el caso de Wright Eagle casi desde el inicio de la competición. Esto se traduce no solo en una mejora continua de las cualidades de sus equipos, sino también en la capacidad de agregar nuevas funcionalidades que los mantengan en puestos competitivos año tras año.

4.2. Recomendaciones para trabajos futuros

Los resultados obtenidos con el *coach* implementado son una mejora considerable sobre el equipo *Agent2D Base*, pero aún así es posible enriquecer más el equipo JEMV.

El uso de las predicciones realizadas por los árboles durante el partido por parte de los jugadores es un área importante que debe mejorarse. Actualmente en

el momento en el que los jugadores deciden realizar una acción, esta es previamente pasada por el árbol de decisión para decidir si será o no exitosa; en caso de que no lo sea, el jugador se queda con el balón hasta el siguiente ciclo. Esto se podría mejorar planteando alternativas adicionales de acciones a realizar en caso de que la inicial sea rechazada por el árbol.

Hacer uso de una mayor cantidad de formaciones para el equipo JEMV, que no necesariamente sean obtenidas de otros equipos que las hagan públicas, permitiría al equipo tener un repertorio más grande de estrategias para contrarrestar oponentes con mayor eficiencia. De la misma forma, tomar en cuenta más información del estado del partido, o agregar más condiciones específicas para generar cambios en la formación de JEMV, podría mejorar el desempeño del equipo si estas sirven para contrarrestar al equipo oponente con mayor eficacia.

Es posible que el método para identificar las formaciones del equipo oponente pueda mejorarse para obtener mayor información que no solo se limite a recabar el esquema táctico oponente. Teniendo una matriz para cada jugador, la cual contiene la información sobre las áreas que el jugador se movió más durante el partido, es posible tener una idea de cómo se mueven los jugadores y hacer uso de esto durante el partido, en vez de limitarse a identificar el área de mayor actividad para dicho jugador.

El método para identificar formaciones tiene una debilidad. Esta se muestra específicamente al usar muchos *logs* de partidos anteriores de equipos que cambian con mucha frecuencia de formación, la cual resulta en una formación promedio un poco más dificil de clasificar. En tiempo real, al jugar contra equipos así, el problema no se presenta ya que hay menos información y menos tiempo para que dicho equipo cambie de formación. Esto permite identificarla mejor. Dicho esto, es un método que se puede mejorar para obtener con mayor exactitud la formación del oponente incluso en caso de que cambie con mucha frecuencia.

Siendo *Agent2D Base* el equipo base de al menos el 83,3 % de los participantes de la Liga de Simulación 2D de RoboCup para el año 2013, cuesta entender que aún no exista algún tipo de documentación que guíe a los investigadores que quieren empezar a incursionar en el área, para dar sus primeros pasos y permitirles centrarse en la generación de nuevos conocimientos. Este problema puede estar

motivado por el hecho de que RoboCup, más allá de permitir avanzar en las investigaciones del área de Inteligencia Artificial y sistemas multi-agentes, no deja de ser una competencia, en la que cada equipo busca ser el ganador año tras año. Por esta razón, se recomienda realizar una labor de documentación más exhaustiva, tanto de los equipos futuros del Grupo de Inteligencia Artificial de la Universidad (GIA), como del software disponible para la comunidad de RoboCup.

Referencias

Akiyama, H., Nakashima, T., Henrio, J., Henn, T., Tanaka, S., Nakade, T., y Fukushima, T. (2016). *HELIOS2016: Team Description Paper.* Fukuoka University, Fukuoka, Japan. Descargado 18/03/2017, de `http://chaosscripting.net/files/competitions/RoboCup/WorldCup/2016/2DSim/tdps/TDP_HELIOS.pdf`

Asali, E., Valipour, M., Afshar, A., Asali, O., Katebzadeh, M., Tafazol, S., ... Mohammadi, M. (2016). *Shiraz soccer 2d simulation team description paper 2016.* Shiraz University, Shiraz, Fars, Iran. Descargado 18/03/2017, de `http://www.robocup2016.org/media/symposium/Team-Description-Papers/Simulation2D/RoboCup_Sim2D_TDP_Shiraz.pdf`

Behnke, S., Veloso, M., Visser, A., y Xiong, R. (2014). Robocup 2013: Robot world cup xvii. *Lecture Notes in Computer Science 8371 Lecture Notes in Artificial Intelligence*.

Berman, E., y Wang, X. (2011). *Essential statistics for public managers and policy analysts.* CQ Press.

Bieger, J., Thórisson, K. R., y Garrett, D. (2014). Raising ai: Tutoring matters. *Artificial General Intelligence: 7th International Conference, AGI 2014, Quebec City, QC, Canada, August 1-4, 2014. Proceedings*, 1–10.

Bishop, C. M. (2006). *Pattern recognition and machine learning (information science and statistics).* Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Breiman, L., H. Friedman, J., A. Olshen, R., y J. Stone, C. (1984). *Classification and regression trees.* Chapman & Hall, New York.

Chen, M., Dorer, K., Foroughi, E., Heintz, F., Huang, Z., Kapetanakis, S., ... Yin, X. (2003, February). Users manual: Robocup soccer server — for soccer

server version 7.07 and later [Manual de software informático]. Descargado de `http://helios.hampshire.edu/jdavila/cs278/virtual_worlds/robocup_manual-20030211.pdf`

Dean, T., Allen, J., y Aloimonos, Y. (1995). *Artificial intelligence: Theory and practice*. Redwood City, CA: Benjamin/Cummings.

de Boer, R., y Kok, J. (2002). *The incremental development of a synthetic multi-agent system: The uva trilearn 2001 robotic soccer simulation team.*

Fathzadeh, R., Mokhtari, V., Mousakhani, M., y Shahri, A. M. (2006). Coaching with expert system towards robocup soccer coach simulation. *RoboCup 2005: Robot Soccer World Cup IX*, 488–495.

Federation, R. (2017). *Soccer Simulation League - Robocup Federation Wiki.* Descargado 18/03/2017, de `http://wiki.robocup.org/wiki/Soccer_Simulation_League`

Habibi, J., Chiniforooshan, E., HeydarNoori, A., Mirzazadeh, M., Safari, M. A., y Younesy, H. R. (2002). Coaching a soccer simulation team in robocup environment. *EurAsia-ICT 2002: Information and Communication Technology: First EurAsian Conference Shiraz, Iran, October 29–31, 2002 Proceedings*, 117–126.

Haykin, S. (1998). *Neural networks: A comprehensive foundation* (2nd ed.). Upper Saddle River, NJ, USA: Prentice Hall PTR.

He, H., Daume, H., y Jason, E. (2012). Imitation Learning by Coaching. *Advances in Neural Information Processing Systems 25*, 3158–3166.

Iglesias, J. A., Ledezma, A., y Sanchis, A. (2009). CAOS coach 2006 simulation team: An opponent modelling approach. *Computing and Informatics*, *28*(1), 57–80.

Javan, M., Ramezanzadeh, A., Kashi, M., Mohamadi, S., y Pashaeehir, A. (2016). *Hermes: Soccer 2D Simulation Team Description Paper 2016.* Department of Computer Science and Information Technology, AllameHelli Highschool, Tehran, Iran. Descargado 18/03/2017, de `http://chaosscripting.net/files/competitions/RoboCup/WorldCup/2016/2DSim/tdps/TDP_HERMES.pdf`

Karimi, M., y Ahmazadeh, M. (2014). Mining robocup log files to predict own and opponent action. *International Journal of Advanced Research in Computer Science*, *5*(6).

Kuhlmann, G., Knox, W. B., y Stone, P. (2006, July). Know thine enemy: A champion RoboCup coach agent. *Proceedings of the Twenty-First National*

Conference on Artificial Intelligence, 1463–68.

Kurose, J. F., y Ross, K. W. (2012). *Computer networking: A top-down approach* (6th ed.). Pearson.

Lau, N., Seabra Lopez, L., Corrente, G., Filipe, N., y Siqueira, R. (2010). Robot team coordination using dynamic role and positioning assignment and role based setplays. *Mechatronics*, *21*, 445–454.

Li, X., Chen, R., y Chen, X. (2015). *WrightEagle 2D Soccer Simulation Team Description 2015.* Department of Computer Science, University of Science and Technology of China. Descargado 18/03/2017, de `http://chaosscripting.net/files/competitions/RoboCup/WorldCup/2015/2DSim/tdps/WrightEagle_TDP.pdf`

Mitchell, T. M. (1997). *Machine learning.* McGraw-Hill Science/Engineering/Math.

RoboCup. (2017). *About Robocup.* Descargado 10/03/2017, de `http://www.robocup.org/objective`

Russell, S. J., y Norvig, P. (2003). *Artificial intelligence: A modern approach* (2.ª ed.). Pearson Education.

Taylor, M. E., Carboni, N., Fachantidis, A., Vlahavas, I. P., y Torrey, L. (2014). Reinforcement learning agents providing advice in complex video games. *Connect. Sci.*, *26*(1), 45–63.

Wackerly, D. D., Mendenhall III, W. M., y Scheaffer, R. L. (2002). *Mathematical statistics with applications* (sixth edition ed.). Duxbury Advanced Series.

Wooldridge, M. (2009). *An introduction to multiagent systems* (2nd ed.). Wiley Publishing.

Apéndice A

Glosario de términos

Agent2D Base. Es un equipo base para la liga de simulación de fútbol 2D de RoboCup creado por el equipo Helios.

Árbol de decisión. Método de aprendizaje de máquinas que permite aproximar funciones objetivo con valores discretos, a partir de un árbol de búsqueda que retorna ua decisión final.

BeforeKickOff. Modo de juego provisto por el servidor que indica que se va a realizar un saque desde la mitad de la cancha, bien sea porque ocurrió un gol o porque es el saque inicial de alguno de los dos tiempos del partido.

Ciclo de juego. Medida de tiempo utilizada por el servidor de RoboCup. Un ciclo equivale a 100ms.

Coach. En RoboCup es un cliente privilegiado utilizado para asistir a los jugadores, puesto que es el único agente que recibe datos libre de ruidos del estado del juego.

CornerKick. Modo de juego de partido adoptado cuando se debe ejecutar un tiro de esquina.

Drible. Acción del fútbol que consiste en que el dueño de la pelota se mueve dentro del campo junto con el balón.

Dueño de la pelota. Jugador que se encuentra más cerca del balón en determinado ciclo, solo si la distancia entre él y la pelota es menor a 0.05 unidades.

Fedit2. Software que permite crear archivos de formación. Estos son usados por los equipos para hacer cambios de esquema táctico durante los partidos.

FreeKick. Modo de juego que indica que se va a producir un tiro libre por parte de uno de los equipos.

Función de activación. Es una función utilizada en redes neuronales para limitar el rango de valores de respuesta generada por cada neurona, generalmente entre [0,1] o [-1,1].

Genius. Equipo desarrollado por la escuela secundaria Ghazal (Shiraz) en Irán y basado en *Agent2D Base*.

GoalieCatch. Modo de juego adoptado cuando un guardameta atrapa el balón con las manos.

GoalieKick. Modo de juego adoptado cuando va a ocurrir un saque de arco por parte del guardameta de un equipo.

Helios. Equipo desarrollado por un grupo de investigadores de la Universidad de Fukuoka, en Fukuoka, Japón. Son los creadores de *Agent2D Base* y del software Fedit2.

Hermes. Equipo basado en *Agent2D Base* y desarrollado por el Departamento de Ciencias de la Computación y Tecnología de Información de la escuela secundaria Allameh Helli en Teherán, Irán.

Jaeger. Equipo desarrollado por la Universidad Tecnológica de Anhui en China y basado en *Agent2D Base*.

JEMV. Equipo implementado en este proyecto partiendo del equipo *Agent2D Base* y ampliando sus funcionalidades para mejorar su desempeño en partidos.

OpenCV. Es una librería *open source* de visión de computadora y aprendizaje de máquina.

Pase. Acción dentro del fútbol que se lleva a cabo cuando un jugador le envía la pelota a un compañero de equipo.

PlayOn. Modo de juego del partido donde el servidor indica que los jugadores se pueden mover libremente por el campo y realizar acciones.

Red neuronal. Procesador paralelo distribuido masivo, compuesto por unidades simples de procesamiento que tienen una disposición natural para almacenar conocimiento producto de experiencias.

Situación del partido. Estado del partido basado en la posición del balón y de los jugadores de ambos equipos durante el modo de juego *PlayOn*.

Situación defensiva. Situación del partido donde algún del equipo oponente puede llegar al balón al menos 2 ciclos antes que cualquier jugador del otro equipo.

Situación normal. Cualquier situación del partido en modo *PlayOn* que no entre dentro de una situación defensiva ni ofensiva.

Situación ofensiva. Situación del partido donde algún jugador de nuestro equipo puede llegar al balón al menos dos ciclos antes que cualquier jugador oponente.

Tiro al arco. Acción que se ejecuta cuando un jugador dispara la pelota hacia la portería del equipo contrario.

Wright Eagle. Equipo desarrollado por el Laboratorio de Sistemas Multiagente de la Universidad de Ciencia y Tecnología de China (USTC).

Apéndice B

Uso de redes neuronales para la clasificación de acciones

Adicionalmente a los modelos empleados en el Capítulo 3 para la clasificación de las acciones de los jugadores, se utilizaron redes neuronales con el fin de poder comparar los resultados obtenidos, verificando si este tipo de modelo de aprendizaje de máquina puede mejorar el porcentaje de acciones clasificadas correctamente.

Al igual que en el caso de los árboles de decisión, las redes neuronales utilizadas para estos experimentos fueron las provistas por la librería OpenCV, dentro de sus funciones de aprendizaje de máquina, en la versión 2.4 para el lenguaje C++. Las redes implementadas en OpenCV son redes *feed-forward* multicapas, que utilizan *backpropagation* como algoritmo de entrenamiento.

En estos experimentos se emplearon los mismos conjuntos de datos que fueron utilizados para las pruebas con árboles de decisión del Capítulo 3, al igual que el proceso de normalización que se les debe aplicar antes de ser usados por los modelos, en este caso las redes neuronales.

Las pruebas fueron realizadas por cada tipo de acción y separadas por equipo oponente (Genius, Helios, Hermes, Jaeger y Wright Eagle). Los experimentos aquí presentados se enfocan en ajustar los parámetros libres de las redes neuronales: el número de neuronas presentes en la capa oculta, el número de épocas usadas para el entrenamiento y la función de activación usada en las neuronas de la red.

El primer parámetro debe ser ajustado para definir la arquitectura de la red. Las neuronas de la capa de entrada están predefinidas por la cantidad de atributos o rasgos utilizados para describir los datos, en este caso 24, y las neuronas de la capa de salida están determinadas por la cantidad de clases que posee el problema, para este caso una neurona de salida es suficiente. Son las neuronas de la capa oculta las que permiten aumentar la dimensionalidad del espacio en el que se trabaja. Se buscará entonces, encontrar la arquitectura de red más simple que mejor logre clasificar los datos utilizados. Para esto se fijó la cantidad de neuronas en las capas de entrada y de salida, y se realizaron pruebas sobre la capa oculta con 10, 20, 30 50 y 80 neuronas.

El segundo parámetro libre que se ajustó fue el número de épocas utilizadas. Este valor determinará el número de veces que se le presentarán los datos a la red durante el proceso de entrenamiento para realizar el ajuste de los pesos. Si se utiliza un número muy elevado de iteraciones el modelo puede presentar problemas por sobreentrenamiento, disminuyendo la capacidad de generalización. Por el contrario, una cantidad de épocas muy pequeña puede ser insuficiente para que la red detecte los patrones necesarios para lograr una buena clasificación. Para las arquitecturas de 10, 20 y 30 neuronas en la capa oculta, se realizaron experimentos con 200 y 500 épocas, mientras que para las redes con 50 y 80 neuronas ocultas, se utilizaron 500 y 800 épocas.

Finalmente, para seleccionar la función de activación se realizaron pruebas usando las funciones provistas por la librería de aprendizaje de máquina de OpenCV. Estas funciones son utilizadas dentro la red neuronal para limitar el rango de valores de la respuesta generada por cada neurona. Generalmente los rangos se limitan a [0,1] o [-1,1]. Para los problemas de clasificación se utilizan funciones de activación no lineales y diferenciables (Bishop, 2006).

La primera función de activación que fue utilizada para la clasificación de cada una de las acciones fue la función sigmoidal simétrica. Esta es la función por defecto de los perceptrones multicapa de OpenCV y su definición viene dada por la siguiente ecuación:

$$f(x) = \frac{\beta * (1 - e^{-\alpha x})}{1 + e^{-\alpha x}} \tag{B.1}$$

donde α se refiere al parámetro de la pendiente de la sigmoidal (Haykin, 1998), mientras que el valor β determinará el rango de la función.

Además de la función sigmoidal, se empleó una función gaussiana definida por la ecuación que se muestra a continuación:

$$f(x) = \beta e^{-\alpha x * x} \tag{B.2}$$

donde el valor de β determinará la altura del pico o punto máximo de la función gaussiana, y el valor de α definirá el ancho de la curva.

Para ambas funciones se utilizaron los valores de $\alpha = 0{,}6$ y $\beta = 1$, recomendados para mejores resultados en la documentación de OpenCV.

Las pruebas para este parámetro se limitaron únicamente a estas dos funciones de activación porque, además de la función identidad, son las únicas soportadas por la librería de aprendizaje de máquina utilizada.

B.1. Clasificación de dribles

Estos experimentos se centraron en evaluar el desempeño de una red neuronal *feed forward* entrenada únicamente con acciones de drible y dribles fallidos del equipo Agent2D Base. A continuación se presentarán únicamente los mejores resultados obtenidos para cada equipo, recordando siempre que fueron realizadas todas las combinaciones de pruebas respecto de la arquitectura de la red, el número de épocas para el entrenamiento y las funciones de activación de las neuronas, explicadas anteriormente.

Los datos extraídos en los partidos contra el equipo Genius constan de 7.310 datos de la clase 'Drible' y 7.310 de la clase 'Drible fallido', que hacen un total de 14.620 datos para la etapa de entrenamiento. De los partidos generados para la fase de pruebas, se extrajeron 3.270 datos. Los mejores resultados obtenidos se encuentran en la Tabla B.1 y son producto de una red neuronal con 24 neuronas en la capa de entrada, 20 en la capa oculta y 1 una en la capa de salida, entrenada

en 500 épocas y con función de activación gaussiana para todas las neuronas. Con los resultados presentes en la tabla se puede concluir que el modelo parece estar teniendo problemas para generalizar, puesto que disminuyó su precisión en casi un 8 % en la fase de prueba respecto de la de entrenamiento.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	14.620	81,881	18,119
Prueba	3.270	73,8226	26,1774

Tabla B.1: Exactitud de una red neuronal para clasificación de dribles contra Genius en fases de entrenamiento y prueba

Es posible que parte del problema de generalización esté siendo causado por una inclinación del modelo hacia los datos de la clase 'Drible'. Los resultados de la Tabla B.2 representan la matriz de confusión para la etapa de entrenamiento del modelo seleccionado. A partir de ellos se puede determinar que existe un 21,08 % de falsos positivos sobre un 15,16 % de falsos negativos. Sin embargo, se seleccionó este modelo sobre otros con mayor porcentaje de aciertos por tener una proporción más equilibrada de clasificaciones exitosas durante el entrenamiento.

K=5	Clasificados 'Drible'	Clasificados 'Drible fallido'
'Drible'	6.202	1.108
'Drible fallido'	1.541	5.769

Tabla B.2: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de dribles con redes neuronales contra Genius

Las redes neuronales para el equipo Helios fueron entrenadas con 14.690 datos, igualmente distribuidos entre ambas clases, y evaluadas con 700 datos. Los resultados del porcentaje de aciertos para ambas fases se encuentran en la Tabla B.3 y fueron generados con una red neuronal de 24 neuronas en la capa de entrada, 20 en la capa oculta y 1 en la de salida, 500 épocas para el entrenamiento y función de activación gaussiana. Con este modelo los datos de entrenamiento fueron clasificados según el 71,9061 %, mientras que los de prueba según el 70,8571 %.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	14.690	71,9061	28,0939
Prueba	700	70,8571	29,1429

Tabla B.3: Exactitud de una red neuronal para clasificación de dribles contra Helios en fases de entrenamiento y prueba

Adicionalmente, en la Tabla B.4 se encuentran los valores asociados a la matriz de confusión de la fase de entrenamiento de la red seleccionada para el equipo Helios. Con estos resultados se puede concluir que la red entrenada para el equipo Helios tiene mayores dificultades en la predicción de los datos de la clase 'Drible', pues la cantidad de falsos negativos es casi el doble que la de falsos positivos.

K=5	Clasificados 'Drible'	Clasificados 'Drible fallido'
'Drible'	4.554	2.791
'Drible fallido'	1.336	6.009

Tabla B.4: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de dribles con redes neuronales contra Helios

Las redes de clasificación de dribles contra el equipo Hermes fueron entrenadas con 9.310 datos y evaluadas con 1.330. Los resultados que se obtuvieron en ambos procesos para el mejor modelo seleccionado se encuentran en la Tabla B.5. Clasificó correctamente el 71,1708 % de los datos de entrenamiento y el 71,5789 % de los datos de prueba. El mejor modelo obtenido fue una red neuronal de 24 neuronas de entrada, 10 en la capa oculta y 1 en la capa de salida, entrenada durante 500 épocas con función de activación sigmoidal.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	9.310	71,1708	28,8292
Prueba	1.330	71,5789	28,4211

Tabla B.5: Exactitud de una red neuronal para clasificación de dribles contra Hermes en fases de entrenamiento y prueba

Con los valores de la Tabla B.6, que representan la matriz de confusión de la fase de entrenamiento, se puede concluir que se generó una mayor proporción de

falsos positivos que negativos con el modelo de clasificación de dribles contra el equipo Hermes.

K=5	Clasificados 'Drible'	Clasificados 'Drible fallido'
'Drible'	3.666	989
'Drible fallido'	1.695	2.960

Tabla B.6: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de dribles con redes neuronales contra Hermes

Para el equipo Jaeger se empleó un conjunto de entrenamiento de 13.400 datos y un conjunto de prueba de 1.986. El mejor modelo generado fue el de una red de 24 neuronas en la capa de entrada, 30 en la capa oculta y 1 en la de salida, entrenada durante 500 épocas y con función de activación gaussiana. Los porcentajes de acierto para dicho modelo se encuentran en la Tabla B.7.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	13.400	74,7463	25,2537
Prueba	1.986	67,8751	32,1249

Tabla B.7: Exactitud de una red neuronal para clasificación de dribles contra Jaeger en fases de entrenamiento y prueba

Además, en la Tabla B.8 se pueden comprobar los valores asociados a la matriz de confusión del mejor modelo seleccionado para el equipo Jaeger. La cantidad de falsos positivos y negativos generada durante la fase de entrenamiento es similar, por lo que no parece existir un sesgo dentro del modelo.

K=5	Clasificados 'Drible'	Clasificados 'Drible fallido'
'Drible'	4.806	1.894
'Drible fallido'	1.490	5.210

Tabla B.8: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de dribles con redes neuronales contra Jaeger

Por último, en la Tabla B.9 se encuentran los valores de la red neuronal que arrojaron los mejores resultados contra el equipo Wright Eagle. De los 16.230 datos

empleados en el entrenamiento, 12.456 fueron clasificados correctamente. Durante la fase de prueba el porcentaje de aciertos disminuyó en un 14,77 % respecto del entrenamiento, por lo que es posible que el modelo tenga problemas de generalización.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	16.230	76,7468	23,2532
Prueba	1.762	61,975	38,025

Tabla B.9: Exactitud de una red neuronal para clasificación de dribles contra Wright Eagle en fases de entrenamiento y prueba

La red seleccionada para este caso está formada por 24 neuronas en la capa de entrada, 10 neuronas en la capa oculta y 1 en la capa de salida. Además, fue entrenada durante 500 épocas y todas las neuronas poseen una función de activación gaussiana.

El problema de generalización con los datos de Wright Eagle puede estar ocasionado por una parcialización del modelo. Aun cuando se seleccionó la red neuronal que, sin disminuir considerablemente el porcentaje de aciertos, tuviera la menor diferencia entre las predicciones correctas de ambas clases, en la Tabla B.10 se puede observar que existe una diferencia de aproximadamente un 10 % entre falsos positivos y negativos generados por la mejor red.

K=5	Clasificados 'Drible'	Clasificados 'Drible fallido'
'Drible'	5.843	2.272
'Drible fallido'	1.502	6.613

Tabla B.10: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de dribles con redes neuronales contra Wright Eagle

B.2. Clasificación de pases

Los experimentos de esta sección fueron realizados para evaluar el desempeño de un árbol de decisión en la clasificación de pases, tanto satisfactorios como fallidos.

Las redes neuronales para la clasificación de pases contra el equipo Genius fueron entrenadas con 10.410 datos, equitativamente distribuidos entre las dos clases. Los resultados alcanzados para la mejor combinación de arquitectura, épocas y función de activación se encuentran en la Tabla B.11. Usando una red de 24 neuronas en la capa de entrada, 30 en la oculta, 1 en la capa de salida, 500 épocas de entrenamiento y función de activación gaussiana se logró clasificar correctamente el 75,7829 % de los datos durante el entrenamiento y el 76,4656 % de los datos de prueba.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	10.410	75,7829	24,2171
Prueba	2354	76,4656	23,5344

Tabla B.11: Exactitud de una red neuronal para clasificación de pases contra Genius en fases de entrenamiento y prueba

Además de los porcentajes totales de aciertos, en la Tabla B.12 se muestran los datos separados en la matriz de confusión de la fase de entrenamiento para Genius. El modelo seleccionado no parece estar parcializado hacia ninguna de las clases. Se obtuvo 26,15 % de falsos positivos y 22,28 % de falsos negativos.

K=5	Clasificados 'Pase'	Clasificados 'Pase fallido'
'Pase'	4.045	1.160
'Pase fallido'	1.361	3.844

Tabla B.12: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de pases con redes neuronales contra Genius

Para el entrenamiento de las redes neuronales contra el equipo Helios fueron utilizados 10.410 datos, mientras que para la evaluación de la calidad del modelo se usaron 550. El modelo que mejores resultados arrojó fue el de una red neuronal de 24 neuronas en la capa de entrada, 10 en la capa oculta y 1 en la capa de salida, entrenada durante 200 épocas y con función de activación gaussiana. Con este modelo se clasificaron correctamente 8.050 datos de entrenamiento, esto es el 74,8607 %, y 422 datos de prueba, que es el 76,7273 %, como así lo reflejan los valores de la Tabla B.13.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	10.410	74,8607	25,1393
Prueba	550	76,7273	23,2727

Tabla B.13: Exactitud de una red neuronal para clasificación de pases contra Helios en fases de entrenamiento y prueba

El modelo de clasificación contra Helios parece estar equilibrado respecto de la predicción por clases. Los datos reflejados en la Tabla B.14 indican que el porcentaje de aciertos para ambas clases difiere en aproximadamente un 2 %.

K=5	Clasificados 'Pase'	Clasificados 'Pase fallido'
'Pase'	3.924	1.281
'Pase fallido'	1.336	3.869

Tabla B.14: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de pases con redes neuronales contra Helios

Las redes neuronales producto de partidos contra Hermes fueron entrenadas con 8.370 datos, y se evaluó su desempeño con 1.170 datos nuevos. De los resultados presentados en la Tabla B.15 se puede concluir que el modelo no parece tener problemas de generalización, ya que el porcentaje de aciertos se mantuvo durante ambas fases. El modelo seleccionado fue el de una red neuronal de 24 neuronas en la capa de entrada, 80 en la oculta y 1 en la de salida, entrenada durante 800 épocas con función de activación sigmoidal.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	8.370	78,0287	21,9713
Prueba	1.170	77,4359	22,5641

Tabla B.15: Exactitud de una red neuronal para clasificación de pases contra Hermes en fases de entrenamiento y prueba

La matriz de confusión obtenida durante el entrenamiento del modelo seleccionado para el equipo Hermes se encuentra en la Tabla B.16. Esta red neuronal parece tener mayores dificultades para predecir correctamente los datos de la clase 'Pase', pues genera un total de 25,27 % de datos de ese conjunto clasificados incorrectamente.

K=5	Clasificados 'Pase'	Clasificados 'Pase fallido'
'Pase'	3.115	1.070
'Pase fallido'	769	3.416

Tabla B.16: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de pases con redes neuronales contra Hermes

Para las redes del equipo Jaeger se usaron 9.640 datos de entrenamiento y 1.386 de prueba, de los cuales con el mejor modelo obtenido se clasificaron correctamente el 74,3672 % y el 75,3247 %, respectivamente, como se muestra en la Tabla B.17. El modelo seleccionado consta de 24 neuronas en la capa de entrada, 80 en la capa oculta y 1 en la capa de salida. Además, fue entrenado durante 500 épocas y las neuronas poseen la función de activación sigmoidal.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	9.640	74,3672	25,6328
Prueba	1.386	75,3247	24,6753

Tabla B.17: Exactitud de una red neuronal para clasificación de pases contra Jaeger en fases de entrenamiento y prueba

Adicionalmente, en la Tabla B.18 se muestran los resultados de la matriz de confusión para dicho modelo. Con estos datos se puede concluir que este modelo parece tener dificultades similares para distinguir elementos de ambas clases.

K=5	Clasificados 'Pase'	Clasificados 'Pase fallido'
'Pase'	3.430	1.390
'Pase fallido'	1.081	3.739

Tabla B.18: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de pases con redes neuronales contra Jaeger

Por último, los modelos para el equipo Wright Eagle se entrenaron con 16.390 datos y fueron evaluados con 2.398, de los cuales, para el mejor modelo generado, se clasificó correctamente el 75,18 % y el 76,4387 %, respectivamente, como se muestra en la Tabla B.19. La red seleccionada está formada por 24 neuronas en

la capa de entrada, 10 en la capa oculta y 1 en la capa de salida, y fue entrenada durante 200 épocas, con función de activación gaussiana.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	16.390	75,18	24,82
Prueba	2.398	76,4387	23,5613

Tabla B.19: Exactitud de una red neuronal para clasificación de pases contra Wright Eagle en fases de entrenamiento y prueba

Tomando como base los valores de la matriz de confusión presente en la Tabla B.20, se puede concluir que el modelo seleccionado parece no encontrarse parcializado, pues genera un 23,39 % de falsos positivos y un 26,25 % de falsos negativos.

K=5	Clasificados 'Pase'	Clasificados 'Pase fallido'
'Pase'	6.044	2.151
'Pase fallido'	1.917	6.278

Tabla B.20: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de pases con redes neuronales contra Wright Eagle

B.3. Clasificación de tiros al arco

En esta sección se presentarán los experimentos para la clasificación de los tiros al arco tanto exitosos como fallidos usando redes neuronales.

Las redes neuronales para la clasificación de tiros al arco contra el equipo Genius fueron entrenadas con 740 datos, igualmente distribuidos entre ambas clases, y evaluadas con 272 datos no usados durante el entrenamiento. Los mejores resultados se consiguieron con una red de 24 neuronas en la capa de entrada, 50 en la capa oculta y 1 en la de salida, 500 épocas para el entrenamiento y función de activación sigmoidal. Con este modelo se clasificaron de forma exitosa 595 datos de entrenamiento, que representan el 80,4054 % del total, y 203 datos de prueba, que representan el 74,6324 %.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	740	80,4054	19,5946
Prueba	272	74,6324	25,3676

Tabla B.21: Exactitud de una red neuronal para clasificación de tiros al arco contra Genius en fases de entrenamiento y prueba

En la matriz de confusión presentada en la Tabla B.22 se observa que el modelo parece clasificar mejor los datos de la clase 'Tiro al arco fallido', puesto que se generaron solo el 14,59 % de falsos positivos, sobre el 24,59 % de falsos negativos durante la fase de entrenamiento de la red.

K=5	Clasificados 'Tiro al arco'	Clasificados 'Tiro al arco fallido'
'Tiro al arco'	279	91
'Tiro al arco fallido'	54	316

Tabla B.22: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de tiros al arco con redes neuronales contra Genius

Para el equipo Helios se entrenaron redes neuronales de 220 datos; posteriormente fueron evaluadas posteriormente con 66 nuevos. Los resultados del porcentaje de aciertos de estos procesos se encuentran en la Tabla B.23. La precisión en ambas etapas fue de 81,8182 % y constituyen el producto de una red de 24 neuronas en la capa de entrada, 30 en la capa oculta y 1 en la capa de salida, con función de activación sigmoidal y entrenada durante 500 épocas.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	220	81,8182	18,1818
Prueba	66	81,8182	18,1818

Tabla B.23: Exactitud de una red neuronal para clasificación de tiros al arco contra Helios en fases de entrenamiento y prueba

Además de los valores de la precisión total, en la Tabla B.22 se puede ver en detalle el comportamiento de la red por cada clase, a partir de la matriz de confusión de la etapa de entrenamiento. De ahí se concluye que, con los datos utilizados para entrenar, el modelo no se encuentra sesgado hacia ninguna clase.

K=5	Clasificados 'Tiro al arco'	Clasificados 'Tiro al arco fallido'
'Tiro al arco'	91	19
'Tiro al arco fallido'	21	89

Tabla B.24: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de tiros al arco con redes neuronales contra Helios

El modelo de clasificación de tiros al arco para el equipo Hermes fue entrenado con 420 datos y posteriormente evaluado con 208. La red de la que se obtuvo mejores resultados corresponde a una arquitectura de 24 neuronas en la capa de entrada, 10 en la capa oculta, 1 en la capa de salida, entrenada durante 500 épocas y con función de activación sigmoidal. Los resultados asociados a esta red se encuentran en la Tabla B.25, donde se puede ver que tanto en la etapa de entrenamiento como en la de prueba se superó el 80 % de datos bien clasificados.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	420	85,7143	14,2857
Prueba	208	80,2885	19,7115

Tabla B.25: Exactitud de una red neuronal para clasificación de tiros al arco contra Hermes en fases de entrenamiento y prueba

Además de la precisión dividida por etapas, en la Tabla B.26 se indican los valores obtenidos durante el entrenamiento, divididos por clase, y se concluye que el modelo arroja resultados similares durante la clasificación de ambos conjuntos de datos.

K=5	Clasificados 'Tiro al arco'	Clasificados 'Tiro al arco fallido'
'Tiro al arco'	185	25
'Tiro al arco fallido'	35	175

Tabla B.26: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de tiros al arco con redes neuronales contra Hermes

Los resultados alcanzados por el mejor modelo de redes neuronales para el equipo Jaeger se encuentran en la Tabla B.27. De los 1.200 datos utilizados para entrenar, el 78,5 % fue clasificado correctamente durante el proceso de validación

cruzada del entrenamiento, y de los 552 empleados para la evaluación de la generalización del modelo, el 73,7319 % fue clasificado de forma exitosa. Estos resultados corresponden a los de una red de 24 neuronas en la capa de entrada, 30 en la oculta y 1 en la de salida. La red fue entrenada durante 500 épocas con función de activación sigmoidal.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	1.200	78,5	21,5
Prueba	552	73,7319	26,2681

Tabla B.27: Exactitud de una red neuronal para clasificación de tiros al arco contra Jaeger en fases de entrenamiento y prueba

Asimismo, en la Tabla B.28 se encuentra la matriz de confusión asociada al mejor modelo seleccionado. A partir de los datos allí exhibidos se observa que la red neuronal no parece tener dificultades diferentes para la predicción correcta de datos de ambas clases.

K=5	Clasificados 'Tiro al arco'	Clasificados 'Tiro al arco fallido'
'Tiro al arco'	474	126
'Tiro al arco fallido'	132	468

Tabla B.28: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de tiros al arco con redes neuronales contra Jaeger

Finalmente, las redes neuronales empleadas para la clasificación de tiros al arco contra el equipo Wright Eagle fueron entrenadas con 430 datos y evaluadas posteriormente con 208. Los resultados obtenidos para el mejor modelo se encuentran en la Tabla B.29. Tanto en la etapa de entrenamiento como en la de prueba se superó el 82 % de datos clasificados correctamente. Esta red neuronal contiene 24 neuronas en la capa de entrada, 50 en la capa oculta y 1 en la de salida. Además, fue entrenada durante 800 épocas y con función de activación sigmoidal.

Fase	Total datos	% Correctos	% Incorrectos
Entrenamiento	430	82,3256	17,6744
Prueba	208	82,6923	17,3077

Tabla B.29: Exactitud de una red neuronal para clasificación de tiros al arco contra Wright Eagle en fases de entrenamiento y prueba

La matriz de confusión asociada al modelo descrito antes se encuentra en la Tabla B.30. De estos datos se puede concluir que la red tuvo más dificultades para clasificar los datos asociados a la clase 'Tiro al arco', que generaron un total de 23,26 % de falsos negativos.

K=5	Clasificados 'Tiro al arco'	Clasificados 'Tiro al arco fallido'
'Tiro al arco'	165	50
'Tiro al arco fallido'	26	189

Tabla B.30: Matriz de confusión correspondiente a la fase de entrenamiento de clasificación de tiros al arco con redes neuronales contra Wright Eagle

B.4. Comparación de resultados con Árboles de Decisión

Se tomó la decisión de utilizar modelos de redes neuronales para la clasificación de acciones con la finalidad de determinar si los problemas encontrados en el entrenamiento de los árboles de decisión estaban siendo ocasionados por el algoritmo como tal de aprendizaje de máquina.

En esta sección se presentarán en modo comparativo los resultados obtenidos por ambos modelos durante la fase de entrenamiento. Se toma únicamente esta fase para la comparación puesto que el bajo desempeño de los modelos no parece estar siendo ocasionado por un problema de sobreentrenamiento, sino de los datos con los que se entrenan. Los resultados completos para las fases de entrenamiento y de prueba se encuentran en el Capítulo 3, para los árboles de decisión, y en las secciones anteriores de este Apéndice, para las redes neuronales.

En la figura B.1 se encuentra la cantidad de datos de dribles clasificados correctamente por ambos modelos, separado por equipos. Se observa que los resultados arrojados por las redes neuronales son similares a los de los árboles en todos los casos, superando únicamente el porcentaje de aciertos para el equipo Genius en un 8,3106 %.

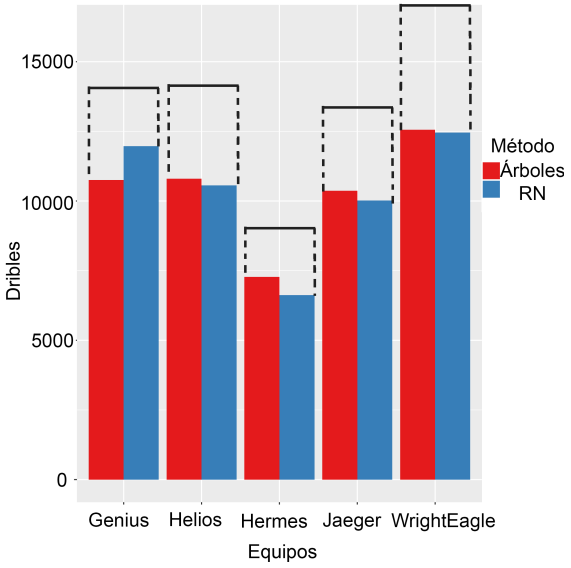


FIGURA B.1: Cantidad de datos bien clasificados por los árboles de decisión y redes neuronales para dribles en fase de entrenamiento

Para la clasificación de pases se presentan en la figura B.2 la cantidad de aciertos obtenidos, separado por equipos. Al igual que en los modelos de dribles, en este caso los resultados son muy similares entre las redes neuronales y los árboles de decisión. Las redes neuronales clasificaron mejor los datos, pero tampoco logró superarse la barrera de 80 % de aciertos.

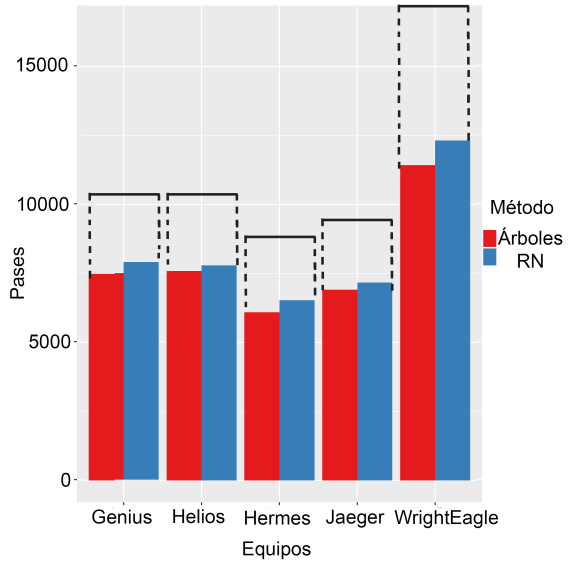
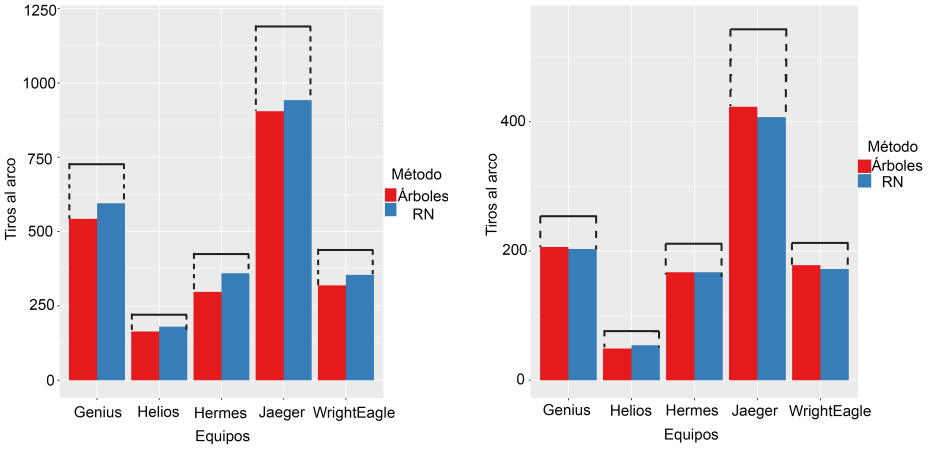


FIGURA B.2: Cantidad de datos bien clasificados por los árboles de decisión y redes neuronales para pases en fase de entrenamiento

Finalmente, en el caso del problema de clasificar tiros al arco, la tendencia de los resultados parece coincidir con la de los dribles y pases, como se muestra en la figura B.3(a). Las redes neuronales generaron resultados ligeramente mejores que los árboles de decisión para clasificar este tipo de datos, llegando a un máximo de 85 % de aciertos en la fase de entrenamiento. Sin embargo, los resultados de las redes neuronales al presentárseles datos nuevos son inferiores a los de los árboles de decisión. Esto queda en evidencia con la gráfica presente en la figura B.3(b), donde las redes neuronales en fase de prueba tienen un desempeño menor que el obtenido en la fase de entrenamiento, incluso menor que el generado por los árboles de decisión.



(a) Fase de entrenamiento

(b) Fase de prueba

FIGURA B.3: Cantidad de datos bien clasificados por los árboles de decisión y redes neuronales para tiros al arco

En conclusión, como se mencionó en el Capítulo 4, aunque las redes neuronales tampoco dieron los resultados esperados para un clasificador, el hecho de que diera resultados similares a los obtenidos por los árboles de decisión hace pensar, como se menciona en el Capítulo 4, que el problema reside en cómo se está modelando el problema con los conjuntos de datos y no en el algoritmo utilizado.

Apéndice C

Pruebas adicionales para contrarrestar estrategias de oponentes

Los métodos para contrarrestar las estrategias de equipos oponentes mostrados en este documento hacen uso de la información del estado del partido y de la formación del oponente para definir la que utilizará JEMV con el fin de contrarrestarlos. Actualmente las condiciones que se usaron para generar cambios en la formación de JEMV son las que dieron mejores resultados en las pruebas, pero no tienen ningún tipo de base en el futbol real, es decir, son condiciones que probablemente no funcionarían correctamente aplicadas a un equipo de fútbol real. Además, las formaciones usadas por JEMV, como se mencionó anteriormente, son formaciones que están disponibles para uso público y fueron creadas por el equipo Genius.

En esta sección se mostrarán los resultados de pruebas estadísticas aplicadas al equipo JEMV haciendo uso de la funcionalidad de detección de formaciones oponentes, pero realizando cambios en los métodos para contrarrestar las estrategias oponentes. En uno de los casos se comparará el desempeño del equipo JEMV con las mismas condiciones para cambiar de formación usadas anteriormente, pero se emplean unas formaciones creadas en Fedit2 para esta investigación específicamente en vez de las de Genius. En el otro caso, se hará uso de las formaciones de Genius, pero con un conjunto de condiciones para cambiar de formación con

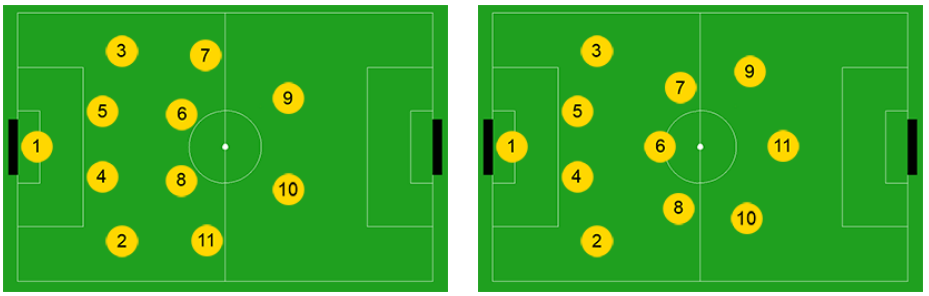
mayor base en el fútbol real. Todo esto para mostrar que los resultados obtenidos en el proyecto para esta funcionalidad provienen de una combinación del uso de las formaciones de Genius con los condicionales aplicados que dieron mejores resultados, y la funcionalidad de detección de formaciones. Específicamente, el uso de las condiciones correctas para cambiar de formación tiene un gran efecto en el desempeño del equipo, como se observará a continuación.

Para los resultados que se mostrarán de seguidas se empleará la misma metodología usada en la sección de experimentos y resultados de este documento. Para las pruebas se tomaron en cuenta 250 partidos de *Agent2D Base*, 50 contra cada uno de los equipos oponentes mencionados anteriormente, los cuales serán considerados los datos de control, y otros 250 partidos del equipo JEMV contra los mismos equipos como los datos experimentales. Se hará uso de pruebas de tipo t de Student para verificar si hay una desigualdad estadísticamente significativa en la media de la diferencia de goles total de ambos grupos, y el α utilizado para las pruebas será de 0,01.

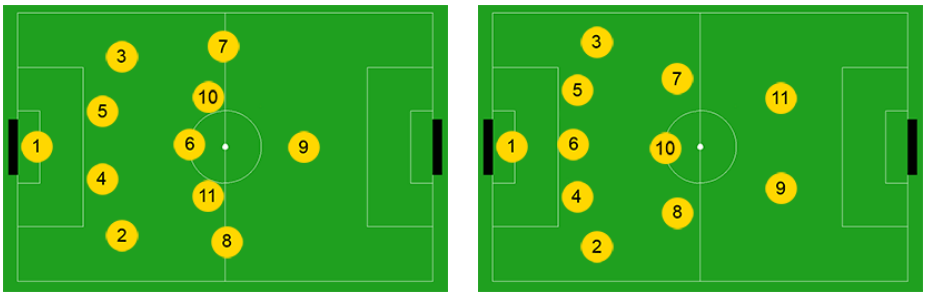
C.1. Pruebas con formaciones creadas en Fedit2

Los experimentos realizados en esta sección comparan el desempeño del equipo *Agent2D Base* con el de JEMV haciendo uso de la funcionalidad de detección de formaciones pero utilizando formaciones creadas en Fedit2 para este experimento con el objetivo de intentar mejorar el desempeño obtenido con las formaciones de Genius. Las condiciones para los cambios de formación son las mismas mencionadas en la sección de implementación.

Las formaciones que se crearon en Fedit2 para este experimento fueron versiones nuevas de 4-4-2 y 4-3-3, y algunas formaciones que Genius no poseía, en específico, 4-2-3-1, 4-5-1 y 5-3-2, todas con sus respectivas versiones asociadas a situaciones ofensivas, defensivas y normales. Dado que las formaciones de Genius públicas tenían pocos esquemas defensivos, se decidió agregar estas últimas tres para compensar. En la figura C.1 se muestran las formaciones mencionadas.



(a) Formación 4-4-2 (b) Formación 4-3-3



(c) Formación 4-5-1 (d) Formación 5-3-2

(e) Formación 4-2-3-1

FIGURA C.1: Formaciones creadas en Fedit2 para este experimento.

De la misma forma, los condicionales para cambio de formación de JEMV se adaptaron a estas nuevas formaciones. Los condicionales utilizados fueron los siguientes:

- Cuando el ciclo actual del partido es mayor o igual a 4.200 (es el último tercio del segundo tiempo):
  - Si el oponente está ganando por menos de dos goles, utilizar 4-3-3.
  - Si el oponente está ganando por dos goles o más, utilizar 4-4-2.

  - Si el oponente está perdiendo y su formación tiene más delanteros que mediocampistas, utilizar 5-3-2.
  - Si el oponente está perdiendo y su formación tiene menor o igual cantidad de delanteros y mediocampistas, utilizar 4-5-1.

- Cuando el ciclo del partido es menor a 4.200:
  - Si la formación del oponente es 4-3-3, se utiliza 4-2-3-1.
  - Si el oponente tiene más delanteros que mediocampistas, usar 4-4-2.
  - Si el oponente tiene 5 o más mediocampistas, utilizar 5-3-2.
  - Si el oponente tiene menos de 5 mediocampistas, utilizar 4-5-1.

La tabla C.1 muestra los resultados de la prueba t de Student aplicada sobre el grupo de control y el experimental. Para esta prueba se tomó como hipótesis nula que la desigualdad entre las medias de la diferencia de goles de ambos conjuntos no es estadísticamente significativa y, como hipótesis alternativa, que sí existe desigualdad estadísticamente significativa entre las medias.

Grupo	Media de diferencia de goles	Diferencia total de goles	Goles a favor	Goles en contra	p-valor
Control	-4,816	-1.204	222	1.426	<2,2e-16
Experimental	-12,40	-3.100	162	3.262	

Tabla C.1: Resultados de la prueba t de Student haciendo uso de formaciones creadas en Fedit2

Se puede observar que existe un aumento estadísticamente significativo en las medias de las diferencias de goles del grupo de control (M=-4,81,SD=3,70) y el experimental (M=-12,40,SD=5,65); t(429,855)=17,73,p<2,2e-16. A partir de los resultados obtenidos se acepta la hipótesis alternativa y se considera el uso de las formaciones creadas en Fedit2 como un deterioro estadísticamente significativo al desempeño del equipo.

La diferencia de goles total para el grupo de control es de -1.204 y para el experimental es de -3.100, lo cual representa un aumento de 1.896 en la diferencia de goles y, por lo tanto, un deterioro sustancial en el desempeño del equipo. Este deterioro se presenta sobre todo en un aumento en los goles en contra, pero no se limita a esto ya que los goles a favor también se ven reducidos. A causa de estos resultados se decidió mantener el uso de las formaciones públicas de Genius en el equipo.

C.2. Pruebas con condicionales con base futbolística

Los experimentos realizados en esta sección comparan el desempeño del equipo *Agent2D Base* con el de JEMV, haciendo uso de la funcionalidad de detección de formaciones y empleando las formaciones publicadas por el equipo Genius pero con un conjunto de condiciones distinto a la hora de realizar cambios de formación en el equipo JEMV. El conjunto de condiciones usado para estas pruebas fue el que mejor resultados dio después del conjunto empleado en la sección de desarrollo e implementación. El objetivo de esto es mostrar que un cambio en las condiciones puede causar un gran cambio en el desempeño del equipo, incluso haciendo uso de las mismas formaciones. Las condiciones adoptadas para estas pruebas fueron las siguientes:

- Cuando el ciclo actual del partido es mayor o igual a 4.200 (es el último tercio del segundo tiempo):
  - Si el oponente está ganando por menos de dos goles, utilizar 4-3-3.
  - Si el oponente está ganando por dos goles o más, utilizar 4-1-2-3.
  - Si el oponente está perdiendo, utilizar 5-4-1.
- Cuando el ciclo del partido es menor a 4.200:
  - Si la formación del oponente es 4-3-3, se utiliza 4-3-3.
  - Si el oponente tiene más delanteros que mediocampistas o más de 5 mediocampistas, usar 5-4-1.

- Si no se cumple ninguna de las dos anteriores, utilizar 4-4-2.

La tabla C.2 muestra los resultados de la prueba t de Student aplicada sobre el grupo de control y el experimental. Para esta prueba se tomó como hipótesis nula que la desigualdad entre las medias de la diferencia de goles de ambos conjuntos no es estadísticamente significativa y, como hipótesis alternativa, que sí existe desigualdad estadísticamente significativa entre las medias.

Grupo	Media de diferencia de goles	Diferencia total de goles	Goles a favor	Goles en contra	p-valor
Control	-4,816	-1.204	222	1.426	0,066
Experimental	-4,23	-966	161	1.127	

Tabla C.2: Resultados de la prueba t de Student haciendo uso de condicionales con base futbolística

En la tabla anterior se puede ver que existe una reducción en la diferencia total de goles, pero la reducción de las medias de las diferencias de goles del grupo de control (M=-4,81,SD=3,70) y el experimental (M=-4,23,SD=3,15); t(473,819)=-1,84,p=0,066, no es estadísticamente significativa, esto se debe a que el p-valor obtenido en las pruebas es mayor al α utilizado de 0,01. Por esto se acepta la hipótesis nula de que la desigualdad entre las medias de la diferencia de goles de ambos equipos no es estadísticamente significativa.

Como se puede observar, el uso de otro conjunto de condicionales que afectan los cambios de formación de JEMV empeoró el desempeño del equipo. A pesar de que los resultados son mejores que los del grupo de control, la diferencia no es estadísticamente significativa como para justificar el uso de que dichas condiciones tengan base en el futbol real.