Protocolo Colaborativo en un Sistema Multi-Agentes BDI para Minería de Datos con Particionamiento Vertical

Universidad Veracruzana

TESIS DE MAESTRÍA

Jorge Melgoza Gutiérrez

CENTRO DE INVESTIGACIÓN EN INTELIGENCIA ARTIFICIAL

"Maestría en Inteligencia Artificial"

23 de Enero de 2015

Documento maquetado con TEXIS v.1.0+.

Protocolo Colaborativo en un Sistema Multi-Agentes BDI para Minería de Datos con Particionamiento Vertical

Tesis para obtener el grado de Maestro en Inteligencia Artificial
Maestría en Inteligencia Artificial

Dirigida por el Doctor
Alejandro Guerra Hernández
Codirector: Dr. Nicandro Cruz Ramírez

CENTRO DE INVESTIGACIÓN EN INTELIGENCIA ARTIFICIAL
"Maestría en Inteligencia Artificial"

23 de Enero de 2015

Agradecimientos

Todo lo que una persona puede imaginar,
otros pueden hacerlo realidad.

Julio Verne

- Agradezco al CONACYT por el apoyo económico brindado durante la realización de este proyecto. CVU: 483440.
- Agradezco a la Universidad Veracruzana, por el apoyo durante mi estancia.
- Agradezco al Dr. Alejandro Guerra Hernández, sus conocimientos y apoyo brindados para la realización de este trabajo.
- Igualmente agradezco al Dr. Nicandro Cruz Ramírez, los conocimientos compartidos que sirvieron de base para este trabajo.
- En especial a mi esposa y a mi hijo, que son la motivación para seguir adelante.
- Agradezco a los compañeros del Departamento de Inteligencia Artificial por sus ideas y comentarios.

Resumen

La ciencia se compone de errores, que a su vez, son los pasos hacia la verdad.

Julio Verne

En la presente tesis se describe un protocolo de aprendizaje colaborativo, modelado e implementado bajo el paradigma de agentes y artefactos, para hacer frente a datos de entrenamiento heterogéneos, es decir, con particionamiento vertical de los ejemplos. Los artefactos encapsulan objetos y métodos de *Weka* con los que se implementa el algoritmo de aprendizaje para inducir árboles de decisión, basado en una versión modificada de J48; mientras que los agentes se encargan de gestionar el flujo del proceso de aprendizaje.

El protocolo propuesto ha sido probado con conocidas bases de datos del *UCI Machine Learning Repository*, comparando la precisión obtenida por la propuesta basada en agentes, contra la tradicional implementación centralizada de J48.

Los resultados muestran que el protocolo de aprendizaje colaborativo logra una precisión equivalente a la obtenida con J48, sin perder la privacidad sobre las instancias de entrenamiento.

Además de la implementación del protocolo de aprendizaje se desarrolló una herramienta para particionar los datos de forma vertical y así poder ejecutar el experimento de forma concurrente.

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Antecedentes

Los algoritmos de minería de datos tradicionales fueron diseñados asumiendo que la información se almacena en una única fuente[28], donde todo el proceso de aprendizaje se realiza en un sitio centralizado. Este punto de vista ha cambiado con el uso generalizado de las redes, en las que se recogen datos de diferentes fuentes y éstos no son necesariamente almacenados en un solo lugar; por los requisitos elevados de almacenamiento o ancho de banda limitado; así también porque las preocupaciones por la privacidad no permiten compartir la información.

En estos casos el enfoque centralizado tradicional no es una opción para hacer un proceso de aprendizaje. La minería de datos distribuida, DDM (Distributed Data Mining) por sus siglas en inglés, puede abordar este tipo de problemas, realizando el análisis de los datos y el proceso de minería de manera distribuida prestando atención a estas limitaciones en los recursos[8]. Los Sistemas Multi-Agente, MAS (Multi-Agent System) por sus siglas en inglés, son capaces de trabajar también de manera distribuida por lo que pueden colaborar con DDM casi naturalmente.

En DDM se abordan principalmente dos tipos de problemas en un sistema de base de datos relacional distribuida: datos homogéneos y datos heterogéneos. El caso de datos homogéneos es también denominado como particionamiento horizontal, aquí el esquema de la base de datos es conocido por todas las partes, sin embargo, nadie tiene todos los ejemplos.

Por otro lado, datos heterogéneos o particionamiento vertical, corresponde al caso donde cada sitio almacena diferentes características pero respecto de los mismos sujetos, un ejemplo son las distintas áreas de un hospital: peso, edad, tipo de sangre; podrían ser obtenidos en un una determinda unidad del hospital, pero la información sobre pruebas de oncología proviene de otra. Es aquí donde cobra importancia el término *privacidad de los datos*, en algunos casos las políticas de privacidad no permiten el intercambio de información

entre unidades aún siendo de una misma organización. En el caso anterior si se desea realizar algún proceso de aprendizaje y obtener un modelo sobre el cáncer, esto no significa que se quiera revelar qué pacientes están enfermos ni de qué.

Para ilustrar mejor lo anterior, podemos observar en la Figura 1.1 cómo se distribuyen los datos cuando hablamos de particionamiento homogéneo u horizontal. La Figura 1.2 muestra un particionamiento heterogéneo o vertical de los datos. En la Figura 1.3 encontramos un ejemplo de un particionamiento híbrido, es decir, datos que se encuentran particionados tanto vertical como horizontalmente.

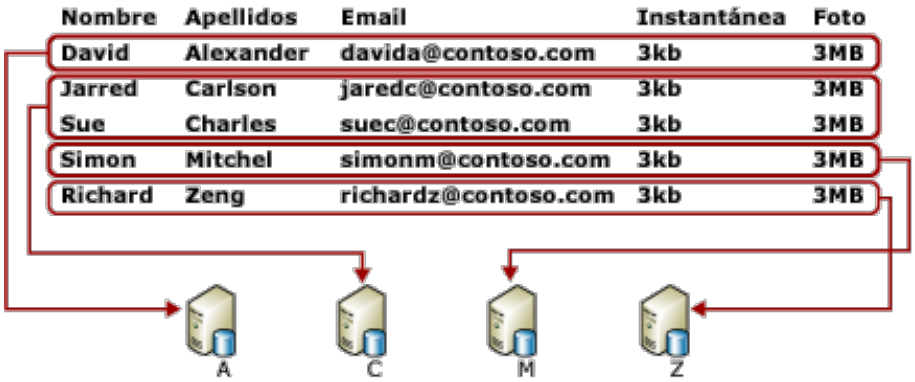

Figura 1.1: Ejemplo de particionamiento horizontal por apellido.

Figura 1.2: Ejemplo de particionamiento vertical.

Figura 1.3: Particionamiento híbrido

Chris Giannella et al[12], proponen una solución donde el trabajo se centra en la construcción eficiente de un árbol de decisión sobre datos distribuidos de forma heterogénea. Para la construcción del árbol de decisión utilizan el algoritmo tradicional ID3[19]. Para que éste funcione de manera distribuida implementan una estrategia basada en el cáculo de la entropía a través de un producto punto. En la Figura 1.4, se observa un ejemplo sencillo de cómo funciona el cálculo mediante el producto punto. Para reducir los costos de comunicación, en lugar de enviar los vectores binarios originales directamente al otro sitio, proyecta los vectores en un espacio dimensional inferior y luego transmite los nuevos vectores a todos los otros sitios; además emplea una estrategia de intercambio de mensajes. De acuerdo a los resultados experimentales mostrados, esta técnica reduce la comunicación por un factor de cinco mientras retiene un 80 % de la precisión original.

Enfocado en la privacidad, Jaideep Vaidya et al. [26] presentan una variante generalizada y preservadora de la privacidad del algoritmo ID3[19] para datos particionados verticalmente y distribuidos a través de dos o más partes. En este enfoque, incluso los meta-datos se ocultan, el esquema (atributos y sus posibles valores) es protegido de ser divulgado, incluso la clase solo es conocida por una de las partes. El algoritmo se implementó sobre Weka[13], con modificaciones que permiten que una instancia sea almacenada en partes distintas, de la misma manera, el proceso interno de clasificación, se modificó para permitir que el control de la ejecución se mueva acorde a las partes que poseen el atributo correspondiente, de acuerdo a la estructura del árbol, aún después de que el modelo es creado los atributos y sus valores no se comparten.

Los Sistemas Multi-Agente son inherentemente distribuidos. Combinar DDM y MAS podría ser una solución viable para ese tipo de problemas. Un

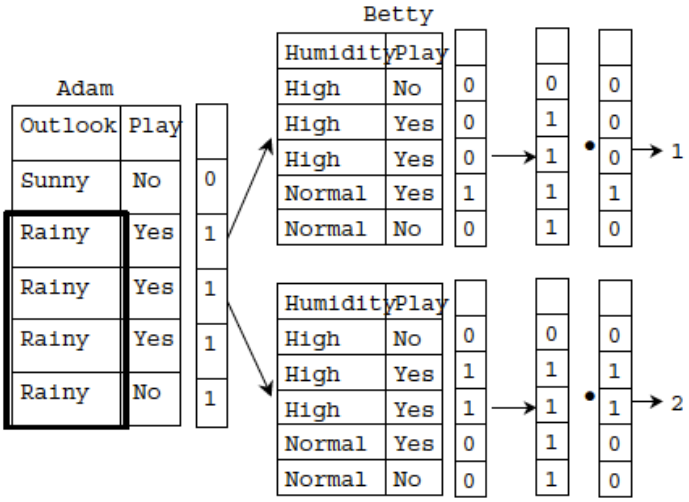

Figura 1.4: Adam envía Betty un vector binario que representa las tuplas con *"Outlook = Rainy"*. Betty construye dos vectores que representan *"Humidity = Normal && Play = Yes"* y *"Humidity = High && Play = Yes"*, respectivamente. El producto punto da el número de tuplas, en toda la base de datos, que satisfacen las restricciones *"Outlook = Rainy && Humidity = Normal && Play = Yes"* y *"Outlook = Rainy && Humidity = High && Play = Yes"*.

enfoque basado en agentes para la inducción de árboles de decisión sobre conjuntos de datos particionados verticalmente, se propone en Sung Baik y Jerzy Bala[3]. Éste trabaja con el algoritmo ID3, original propuesto por Quinlan et al.[19], está diseñado para dos agentes que interactuán a través de un mediador, para construir árboles de decición. Su meta es reducir el ancho de banda de la comunicación entre agentes, buscando maneras de reducir la cantidad de información necesaria para la colaboración de los mismos, a fin de reducir la necesidad de recolectar los datos desde los sitios distribuidos por medio de la aplicación del análisis de los datos en dichos sitios.

Xavier Limón et al[15], proponen una solución para DDM modelada e implementada dentro del paradigma de *Agentes y Artefactos*, en ésta se utiliza el lenguaje de programación orientado a agentes Jason[4] y CArtAgO[23, 22]. En ella se presenta la herramienta denominada JaCa-DDM enfocada al particionamento horizontal de los datos, sin embargo, la metáfora de agentes y artefactos podría ser también empleada para problemas de particionamiento vertical.

Si bien se han empleado agentes inteligentes para problemas de minería de datos con particionamiento vertical, la opción de agentes y artefactos no ha sido explorada, así como un algoritmo de aprendizaje distinto de ID3[19] para la construcción de árboles de decisión.

1.2. Definición del Problema

La minería de datos ha evolucionado del procesamiento centralizado de los datos, a problemas más complejos como el particionamiento horizontal y vertical de los mismos.

El segundo caso, también denominado como datos de entrenamiento heterogéneos, se presenta cuando la información se encuentra almacenada en distintos sitios, cada uno compartiendo en esencia diferentes características de las instancias, pero ya sea debido a los requisitos elevados de almacenamiento o ancho de banda limitado, o a las preocupaciones de privacidad; la información no debe ni puede ser centralizada para su procesamiento. En la Figura 1.5 se puede observar un ejemplo de particionamiento vertical de la información.

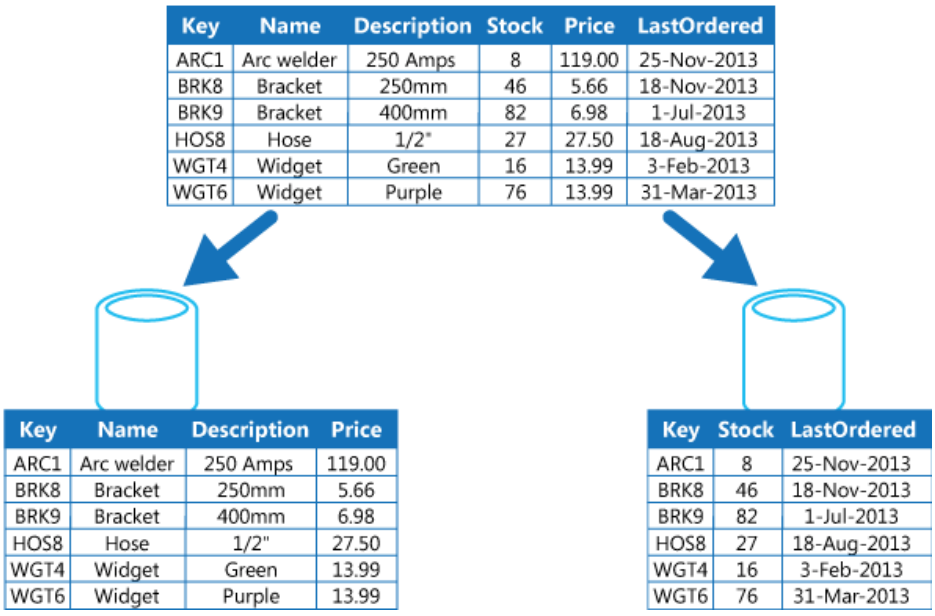

Figura 1.5: Particionamiento Vertical

En la presente tesis se abordará este tipo de problemas, desde la perspectiva de la preservación de la privacidad de los datos para la generación

de árboles de decisión. Si bien es cierto que dentro de la minería de datos distribuida se presenta un compromiso importante entre la certeza de clasificación del modelo y el costo de la inducción del mismo[21], donde el costo de inducción se refiere tanto al costo computacional para la generación del modelo, como al de transmisión de los datos; esta investigación se centrará en la viabilidad del enfoque propuesto para obtener niveles equivalentes, en cuanto a la certeza de la clasificación, comparados con el enfoque centralizado tradicional; particularmente aplicado al problema del particionamiento vertical de los datos, desde la perspectiva de la preservación de la privacidad. Lo anterior debido a que se considera que la certeza en la clasificación tiene una mayor relevancia, además, en el presente trabajo, la comunicación consistirá en transmitir sólo los valores obtenidos de los cálculos estadísticos, no los datos, por lo que la comunicación ya se ha reducido en cierta manera.

1.3. Objetivos

1.3.1. Objetivo General

Diseñar e implementar un protocolo de aprendizaje colaborativo basado en agentes, para problemas de minería de datos con particionamiento vertical, enfocado a la creación de árboles de decisión, que tenga un rendimiento competitivo comparado con el enfoque centralizado tradicional, en términos de la precisión del clasificador.

1.3.2. Objetivos Específicos

- Desarrollar un protocolo de aprendizaje colaborativo basado en agentes para la creación de árboles de decisión.
- Analizar el diseño interno de la implementación de C4.5 realizada en Weka[13], para realizar las modificaciones necesarias, a fin de que el control sobre la generación del árbol de decisión esté a cargo de un agente BDI basado en Jason.
- Complementar la herramienta de JaCa-DDM[15], a fin de desarrollar una aplicación basada en agentes BDI que, mediante técnicas de minería de datos, realice aprendizaje sobre una base de datos cuya información se encuentra particionada de forma vertical, esto es, los atributos de la misma no se encuentran centralizados.
- Producir una hipótesis con un grado de precisión competitivo respecto de la obtenida por el modelo centralizado tradicional.
- Analizar el protocolo de aprendizaje colaborativo, así como el algoritmo de aprendizaje, para estudiar posibles criterios de optimización.

1.4. Contribución

- Un protocolo de aprendizaje colabortativo basado en agentes bajo el esquema de agentes y artefactos.
- Una implementeación del protocolo de aprendizaje colaborativo, desarrollada con los lenguajes orientados a agentes Jason y CArtAgO, así como código reutilizado de Weka.
- Una mejora a la herramienta JaCA-DDM, que le permite ejecutar experimentos donde los datos se particionan de manera vertical.

1.5. Hipótesis

"Un Sistema Multi-Agente que utilice procedimientos de minería de datos para la creación de árboles de decisión en ambientes heterogéneos, obtendrá niveles de precisión competitivos contra el enfoque centralizado tradicional"

1.6. Justificación

El problema del particionamiento vertical en la minería de datos, particularmente en cuanto a la generación de árboles de decisión, ha sido atacado con anterioridad. Sin embargo, los enfoques propuestos se han basado en el algoritmo ID3, originalmente propuesto por Quinlan[19].

En cuanto al uso de agentes inteligentes, Sung Baik y Jerzy Bala[3] proponen una solución basada en agentes, diseñada para dos agentes que interactuán para generar el árbol de desición. El paradigma de agentes y artefactos ha sido empleado con buenos resultados en Xavier Limón et al[15], donde se plantea una solución de minería de datos para ambientes homogéneos.

De acuerdo a la revisión de la literatura especializada sobre particionamiento vertical de los ejemplos, este problema se puede atacar desde dos enfoques: a través de un intermediario cofiable, esto es, un tercero que actúe como facilitador entre las dos partes que participan en la comunicación de los datos, de tal manera que la información no fluya directamente entre los participantes; o mediante mecanismos de encriptación para la transmisión de los datos como en Jaideep Vaidya et al. [26].

Si se pretende usar un intermediario, un artefacto puede cubrir ese rol, ya que dadas sus características, éstos solo revelan determinada información a los agentes que interactúan con ellos.

Se puede observar que la combinación de agentes y artefactos no ha sido probada para el problema del particionamiento vertical de los datos, así como tampoco se ha explorado el comportamiento del algoritmo J48 para la generación de árboles de decisión en ambientes heterogéneos.

1.7. Organización del documento de tesis

El documento se organiza de la siguiente manera:

- Capítulo 2. Sistemas Multi-Agente. Este capítulo presenta conceptos básicos de los Sistemas Multi-Agente.
- Capítulo 3. Protocolo de Aprendizaje Colaborativo. En este capítulo se describen los agentes y artefactos que forman parte del protocolo de aprendizaje, el protocolo y las particularidades derivadas de la implementación.
- Capítulo 4. Resultados y Discusión. Aquí se explican las condiciones en las que se llevaron a cabo los experimentos, además se presentan y discuten los resultados obtenidos donde se compara la estrategia propuesta en contra de la tradicional estrategia centralizada.
- Capítulo 5. Conclusiones y Trabajo Futuro. Este capítulo contiene las conclusiones obtenidas a partir de los resultados y el trabajo futuro derivado de esta tesis.

Capítulo 2

Sistemas Multi-Agente

2.1. Agentes computacionales

La noción de agente no es exclusiva de la IA, otras disciplinas comparten como objeto de estudio a las entidades inteligentes y su comportamiento; sin embargo, a diferencia de la filosofía, la psicología, las neurociencias y demás disciplinas que lo estudian, la IA se enfoca no solamente en la comprensión de dichas entidades, sino en su construcción. Dichas disciplinas aventajan a la IA en el estudio de los llamados agentes inteligentes, pues ésta es relativamente nueva al igual que otras ciencias de la computación, mientras que ciencias como la filosofía son mucho más antiguas, es por ello que son una buena referencia de conocimiento a partir del cual desarrollar los conceptos adoptados por la IA.

El término agente se emplea desde Aristóteles[1] continuando hasta nuestros días por los filósofos, para referirse a una entidad que actúa con un propósito dentro de un contexto social. En el ámbito legal designa a la persona que actúa en beneficio de otra con un propósito específico, bajo la delegación limitada de autoridad y responsabilidad, este se encuentra presente en el derecho Romano y ha sido ampliamente utilizado en economía [16].

En el contexto de las ciencias de la computación, Russell y Norvig[24] definen a la IA como el estudio de agentes que reciben percepciones del entorno y llevan a cabo acciones; para ellos la construcción de agentes racionales constituye la idea central de lo que llaman el enfoque moderno de la IA.

Definir universalmente qué entendemos por agente racional artificial es una tarea difícil, por no decir imposible, pues conlleva definir qué entendemos por inteligencia, lo cual es materia de grandes y acalorados debates. Sin embargo, para ayudar a la comprensión del tema podemos tomar la siguiente definición consensual de agente[27, 24]:

> Un **agente** es un sistema computacional capaz de actuar de ma-

nera **autónoma** para satisfacer sus objetivos y **metas**, mientras se encuentra situado persistentemente en su **medio ambiente**.

Si observamos la Figura 2.1, podemos notar que las **acciones** del agente modifican el medio ambiente y a su vez los cambios en él son **percibidos** por el agente.

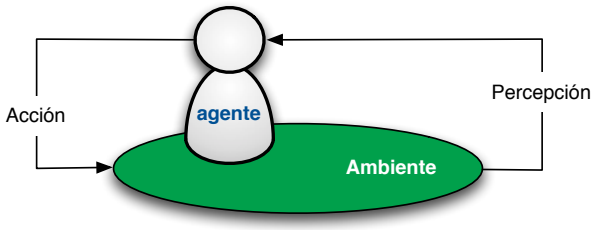

Figura 2.1: Representación de un agente a partir de su interacción con el medio ambiente.

Sin embargo, esto no es suficiente para caracterizar a un agente como inteligente. Un foco con un sensor de movimiento puede cubrir la descripción anterior, percibe el ambiente y actuá en consecuencia encendiendo la luz cuando alguien se encuentra en la misma habitación, pero no debemos considerarlo como inteligente.

Wooldridge y Jennings [27], mencionan características del comportamiento de un agente inteligente:

- Reactividad. Los agentes inteligentes deben ser capaces de percibir su medio ambiente y responder a tiempo a los cambios en él, a través de sus acciones.
- Iniciativa. Los agentes inteligentes deben exhibir un comportamiento orientado por sus metas, tomando la iniciativa para satisfacer sus objetivos de diseño y posiblemente decidiendo qué objetivo atender.
- Sociabilidad. Los agentes inteligentes deben ser capaces de interactuar con otros agentes, posiblemente tan complejos como los seres humanos, con miras a la satisfacción de sus objetivos. Esto incluye la capacidad de comunicarse, negociar y alcanzar acuerdos.

Además de las ya mencionadas, la autonomía es una de las características que definen a un *agente inteligente*. Covrigaru y Lindsay [7] expresan que un agente se percibe como **autónomo** cuando:

1. Su comportamiento está orientado por sus metas y es capaz de seleccionar qué meta va a procesar a cada instante.

2. Su existencia se da en un período relativamente mayor al necesario para satisfacer sus metas.

3. Es lo suficientemente robusto como para seguir siendo viable a pesar de los cambios en el ambiente.

4. Puede interaccionar con su ambiente en la modalidad de procesamiento de información.

5. Es capaz de exhibir una variedad de respuestas, incluyendo movimientos de adaptación fluidos; y su atención a los estímulos es selectiva.

6. Ninguna de sus funciones, acciones o decisiones, está totalmente gobernada por un agente externo.

7. Una vez en operación, el agente no necesita ser programado nuevamente por un agente externo.

2.1.1. Intencionalidad - Agentes BDI (Belief-Desire-Intention)

Muchos de nuestros estados mentales[1] están en cierto sentido dirigidos a objetos o asuntos del mundo. Si tengo una creencia, debe ser una creencia que tal y tal es el caso; si deseo algo, debo tener el deseo de hacer algo, o que algo ocurra, o que sea el caso; si tengo una intención, debe ser la intención de hacer algo; etc. Es esta característica de **direccionalidad** en nuestros estados mentales, lo que muchos filósofos han etiquetado como **Intencionalidad**[25].

Lo que es relevante en la anterior definición, es que los estados mentales Intencionales parecen tener una estructura o prototipo que consiste en una **actitud**, como creer, desear, intentar, etc., que opera sobre el **contenido** del estado, que a su vez está relacionado con algo más allá de si mismo, el objeto hacia el cual apunta. En este sentido, los estados Intencionales son representaciones de **segundo orden**, es decir, representaciones de representaciones. Si además, el contenido de un estado Intencional se puede expresar en forma proposicional, hablamos de una **actitud proposicional**. La Figura 2.2 ilustra estas definiciones.

De acuerdo con Dennett[10, 9], los sistemas Intencionales son por definición, todas y sólo aquellas entidades cuyo comportamiento puede ser explicado o predicho, al menos algunas veces, asumiendo una **postura Intencional**. Tal postura consiste en la interpretación del comportamiento de la entidad en cuestión (persona, animal o artefacto) asumiendo que se trata de un **agente racional** que gobierna su selección de acción considerando sus **actitudes proposicionales**: creencias, deseos, intenciones, etc. La postura

[1] Los estados mentales son por ejemplo: creencias, deseos, intenciones, etc.; así como otros que no son Intencionales como el dolor, entre otros.

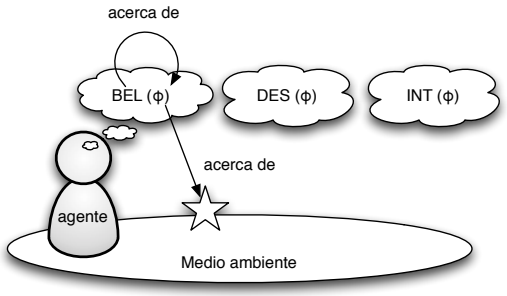

Figura 2.2: Las actitudes proposicionales son representaciones de segundo orden. La proposición ϕ puede ser "la estrella es blanca" y es acerca de la estrella blanca en el medio ambiente. El agente puede tener como actitud creer (BEL), desear (DES) o intentar (INT) esa proposición. Las actitudes son acerca de la proposición, no acerca de la estrella en el medio ambiente.

Intencional puede verse como una **estrategia**, de entre otras posibles, para explicar o predecir el comportamiento de un agente.

Otro concepto fundamental es el *razonamiento práctico* propuesto por Bratman[5]. Son cuatro los supuestos básicos de esta teoría:

- Las intenciones están ligadas a los planes, de hecho son agregados de planes parciales y jerárquicos.
- Somos agentes que planean para contender con nuestra racionalidad acotada y poder decidir ahora qué haremos en el futuro.
- Somos agentes racionales, de forma que nuestros planes y su ejecución dependen de cierta deliberación.
- Una intención no es igual que un deseo, aunque ambos tienen roles motivadores, en realidad una intención conlleva compromiso, mientras que un deseo no.

Los deseos como las intenciones son pro-actitudes, pero mientras los primeros son potenciales influencias de la conducta, las intenciones conducen la conducta. Esto es, los deseos no implican compromiso, ni explican la planeación a futuro; las intenciones si.

Dado el supuesto de que los agentes planean, el comportamiento de éstos está delineado por un proceso de razonamiento práctico. Este tipo de razonamiento está enfocado a realizar acciones basadas en lo que el agente cree y desea, y tiene dos características importantes:

- La deliberación, que consiste en la adopción de intenciones, basada en razones creencia-deseo, es decidir qué metas debe lograr.

- El razonamiento medios-fines que consiste en la determinación de los medios para satisfacer las intenciones. Las salidas de los razonamientos medios-fines son planes.

Bajo estos dos procesos, se pueden distinguir las siguientes características en las intenciones:

- Pro-actividad. Las intenciones pueden motivar el cumplimiento de metas, son controladoras de la conducta.
- Inercia. Las intenciones persisten, es decir, una vez adoptadas se resisten a ser revocadas. Sin embargo, no son irrevocables. Si la razón por la cual se creo la intención desaparece, entonces es racional abandonar la intención.
- Intenciones futuras. Una vez adoptada una intención, ésta restringirá los futuros razonamientos prácticos, en particular el agente no considerará adoptar intenciones incompatibles con la intención previamente adoptada. Es por ello que las intenciones proveen un filtro de admisibilidad para las posibles intenciones que un agente puede considerar.

2.1.2. Planes

Los planes en tanto cursos de acción, son intenciones y, en ese sentido comparten las propiedades de estas: poseen inercia, son controladores de la conducta del agente y sirven como futuras entradas para próximos razonamientos prácticos. Sin embargo, los planes también poseen otras características distintivas.

- Los planes son parciales, no son estructuras completas y estáticas.
- Los planes tienen una estructura jerárquica, contienen razones medios-fines y estas razones tienen un procedimiento ordenado.
- Los planes poseen consistencia interna en el sentido de poder ser ejecutables.
- Los planes son fuertemente consistentes con las creencias del agente.
- Los planes poseen coherencia medios-fines en el sentido de que los subplanes de un plan son coherentes con los fines del plan.

2.2. Medio Ambiente

El medio ambiente para un sistema multi-agentes es el lugar donde uno o varios agentes se ubican, donde llevan a cabo sus acciones y buscan cumplir sus metas. Brooks[6] argumenta que el medio ambiente por excelencia

es el mundo real, proponiendo que todo agente toma una forma robótica. Etzioni[11] al contrario, sostiene que no es necesario que los agentes tengan implementaciones robóticas, los ambientes virtuales como los sistemas operativos y la web, son igualmente válidos.

Una tercera perspectiva acorde al paradigma empleado en este trabajo, es propuesta por Ricci[18, 22], donde la interacción de los agentes con el medio ambiente se concibe como una interfaz, esto es una capa de servicios compuesta por objetos reactivos, más no autónomos ni proactivos, llamados artefactos. Es a través de estos objetos que los agentes pueden percibir su medio ambiente y actuar en consecuencia logrando efectos en él. Estos conceptos se pueden observar en la Figura 2.3, los agentes no interactúan directamente con el ambiente, lo hacen a través de esta capa de objetos que representa una abstracción del ambiente mismo. La figura corresponde a la herramienta JaCa (Jason & CArtAgO), basada en el paradigma de agentes y artefactos, que como se menciono guarda relación con la propuesta de Ricci, se hablará de dicha herramienta en la siguiente sección.

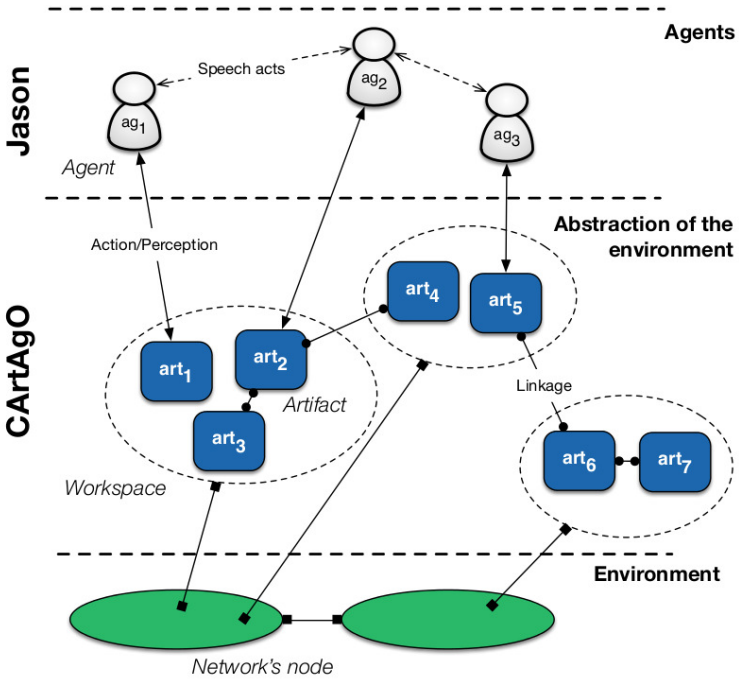

Figura 2.3: Interacción de los agentes con el ambiente a través de la capa de servicios compuesta por un conjunto de artefactos.

Agentes y artefactos pueden organizarse en espacios de trabajo, para

reflejar la topología del ambiente. Esto constituye un meta-modelo que puede aproximar ambientes tanto reales, como virtuales.

El concepto de espacio de trabajo se asemeja al de entorno de trabajo, propuesto por Russell y Norvig[24] para especificar tipos de agente con base en la medida de desempeño utilizada, el ambiente donde el agente está situado, los actuadores del agente y sus sensores. Se le conoce como PEAS por sus siglas en inglés (Performance, Environment, Actuators, Sensors).

En el caso de los espacios de trabajo, sensores y actuadores están encapsulados en los artefactos. Con la ventaja de poder usar cualquier metodología de programación, por ejemplo una orientada a objetos con Java.

2.3. Jason & CArtAgO

JaCa (Jason & CArtAgO) es una herramienta que sirve como marco de trabajo para la programación de Sistemas Multi-Agente. Combina dos tecnologías diferentes, cada una de ellas es bien conocida por sí sola y su desarrollo lleva varios años, por lo que son bastante robustas.

Jason es un intérprete para una versión ampliada de AgentSpeak. Implementa la semántica operacional de ese lenguaje y proporciona una plataforma para el desarrollo de sistemas multiagente, con muchas características personalizables por el usuario.

CArtAgO por su parte, es un marco de trabajo o infraestructura de propósito general, que facilita programar y ejecutar ambientes virtuales para MAS.

Estas dos tecnologías son fuertemente acopladas, el desarrollo de MAS utilizando a Jason para definir a los agentes y a CArtAgO para especificar el ambiente, no requiere de trabajo extra para su funcionamiento en conjunto.

Primero se mencionarán las particularidades del lenguaje de programación orientado a agentes Jason, para después hablar sobre la infraestructura para ambientes basados en artefactos CArtAgO.

2.3.1. Jason

Para la implementación de sistemas basados en agentes, se han desarrollado diferentes lenguajes de programación, para el caso particular de la implementación aquí propuesta se utilizará el lenguaje Jason[4], el cual es un lenguaje de programación orientado a agentes basado en Java que implementa una semántica operacional extendida de AgentSpeak.

El lenguaje está influenciado por el trabajo en arquitecturas BDI y lógicas BDI[20].

Para definir el ambiente en Jason se emplea Java, el trabajo de razonamiento del agente se lleva a cabo en términos de AgentSpeak, lo cual es consistente con el concepto de ambiente propuesto por Ricci[18, 22], y permi-

te cierta separación entre ambiente y agentes. Además de beneficiarse de las bondades como lenguaje de programación, brinda la posibilidad de realizar extensiones por medio de bibliotecas Java.

Dentro de las funcionalidades de Jason se encuentran:

- Comunicación inter-agente basada en actos de habla.
- Anotaciones en las creencias que especifican el origen de la información.
- Anotaciones en las etiquetas de los planes, que pueden ser utilizadas por funciones de selección personalizadas.
- Funciones de selección personalizadas en Java, funciones de confianza y arquitecturas de agente personalizadas.
- Extensibilidad directa mediante *acciones internas* definidas por el usuario en código Java.
- Una noción clara de ambiente multi-agente implementado en Java, el cual puede ser utilizado para simular un ambiente real.

2.3.2. Ambientes Basados en Artefactos, CArtAgO

Un artefacto es una abstracción de primer orden utilizada para el modelado de ambientes computacionales. Cada artefacto representa una entidad del medio ambiente, ofreciendo servicios (operaciones) y estructuras de datos (propiedades observables) a los agentes, a fin de que éstos puedan llevar a cabo sus actividades, las cuales a su vez producen cambios en el ambiente que serán percibidos por medio de los artefactos.

Los artefactos se conciben como dispositivos computacionales orientados a la funcionalidad, los agentes pueden explotar esta funcionalidad con el fin de cumplir con sus actividades individuales y sociales.

CArtAgO[22] (Common ARTifact infrastructure for AGents Open environments) es un marco de trabajo computacional para la Programación de Ambientes basado en el meta-modelo de Agentes y Artefactos como se muestra en la Figura 2.4.

El ambiente se concibe como un conjunto dinámico de entidades computacionales llamadas artefactos, que representan generalmente recursos y herramientas que los agentes trabajando en el mismo ambiente pueden compartir y explotar. Este conjunto de artefactos puede organizarse en uno o múltiples espacios de trabajo, posiblemente distribuidos en diferentes nodos de una red de cómputo. Un espacio de trabajo representa una localidad.

El ambiente es entonces, un conjunto de artefactos a través de los cuales el agente puede percibir el estado de éste y realizar acciones utilizando dichos artefactos.

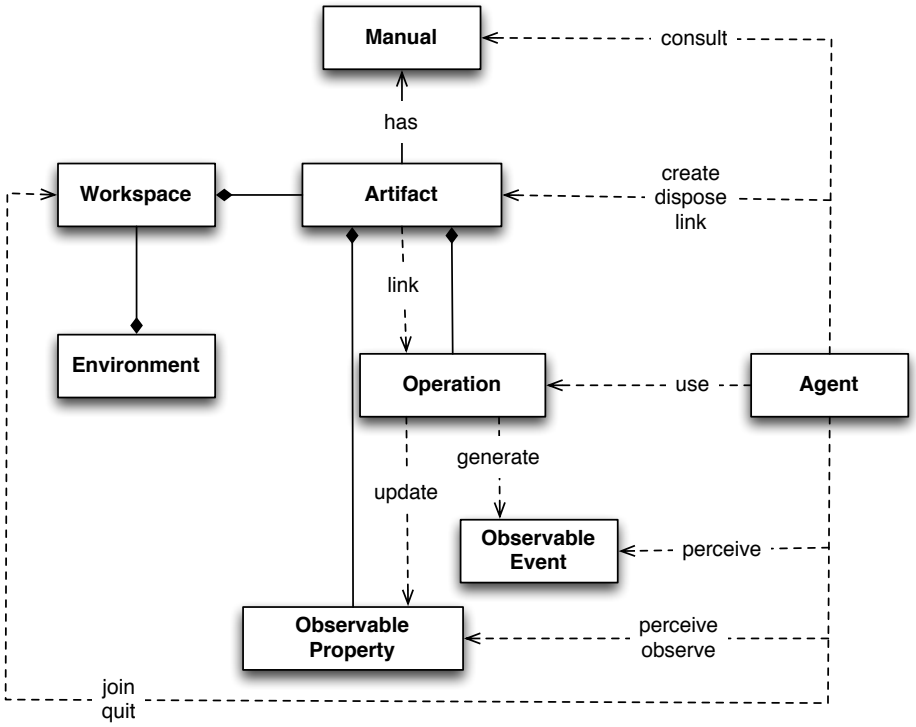

Figura 2.4: Meta-modelo de Agentes y Artefactos.

La infraestructura de CArtAgO está pensada para funcionar en ambientes distribuidos, pudiéndose definir espacios de trabajo que determinan el contexto donde un artefacto existe y por lo tanto puede ser percibido y utilizado.

Capítulo 3

Protocolo de Aprendizaje Colaborativo

En el protocolo de aprendizaje propuesto, participan dos tipos de agentes: *learner* y *worker*; éstos a su vez intereactúan con dos artefactos: *classifier* y *attManager*.

En la Figura 3.1 se puede observar de manera general el funcionamiento del protocolo de aprendizaje.

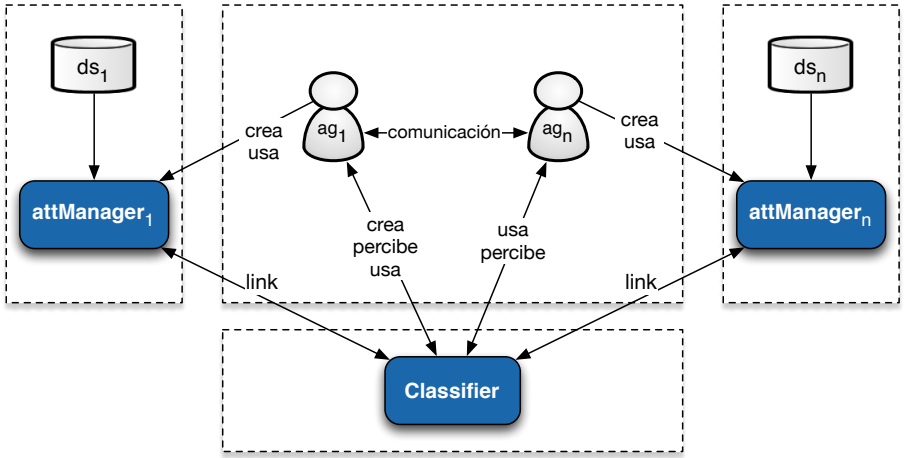

Figura 3.1: Esquema general del protocolo de aprendizaje colaborativo.

Cada agente, actuando como *worker*, posee un artefacto *attManager* propio, con el cual manipula sus datos localmente. A se vez uno de los agentes cumple el rol de *learner*, éste coordina el aprendizaje mediante actos de habla

y señales emitidas por el artefacto *classifier*. La información que es compartida, se envía a través de enlaces entre los artefactos *attManager* y *classifier*. Los agentes y artefactos se explicarán a detalle en las siguientes secciones de este capítulo.

3.1. Agentes

Durante el proceso de aprendizaje, los agentes participantes deben desempeñar dos roles: aprendiz y trabajador. Los agentes fueron diseñados de tal manera, que cualquiera puede ser aprendiz o trabajador, sin embargo para que el orden se mantenga durante el proceso de inducción del árbol de decisión, solo puede haber un aprendiz y uno o varios trabajadores.

Si bien un solo agente podría desempeñarse como aprendiz y trabajador al mismo tiempo, un sistema con un solo agente no requiere compartir datos y por ende no se trataría de un problema de particionamiento de la información.

La forma en que un agente sabe si debe actuar conforme a uno u otro rol, es de acuerdo a su base de conocimientos, cada agente asume que debe actuar como *worker*, a no ser que tengan la creencia de que él es el *learner* del grupo. En esta fase de la investigación, dicha creencia es parte de la base de conocimientos del agente desde que éste se crea, durante todo el experimento el rol de agente *learner* no cambia.

3.1.1. *Learner* - Aprendiz

El agente *aprendiz* en adelante *learner*, es el encargado de controlar el flujo del proceso de aprendizaje, es quien decide por ejemplo, en qué momento se evalúan los atributos.

Cada ciclo de aprendizaje, antes de solicitar información a los demás agentes, evalúa las condiciones de paro del algoritmo de aprendizaje y, de ser el caso, crea los nodos hoja correspondientes.

El *learner* también verifica paso a paso el estatus del nodo actual y por ende el del modelo, es quien determina cuándo hacer *backtrack*, cuándo es necesaria una partición o cuándo el modelo está terminado.

3.1.2. *Worker* - Trabajador

Son aquellos cuyos planes están enfocados en dar respuesta a las peticiones del agente *learner* y reaccionar a las señales del artefacto clasificador (se explicará más adelante).

Cada *worker* evalúa de forma local sus atributos de acuerdo al algoritmo de aprendizaje implementado, para este trabajo en *J48* se calcula la *Ganancia de Información*, cuando así se lo solicita el agente *learner*. Después

de evaluar sus atributos, comunica los resultados al artefacto *classifier*, pero sin divulgar los nombres de los atributos ni sus valores, se comunican números enteros en representación del nombre del atributo, la equivalencia entre el entero y el nombre del atributo sólamente es conocida localmente por el agente.

Otra tarea a cargo de estos agentes es la creación de nuevos nodos. Una vez que la etapa de selección del mejor atributo concluye, por medio de una señal del artefacto *classifier*, se le notifica al *worker* respectivo qué atributo fue seleccionado, dicho agente creará el nodo nuevo con el atributo ganador y lo enviará al artefacto *classifier* para que sea agregado al modelo.

3.2. Artefactos

Para el proceso de aprendizaje hay dos artefactos principales: *classifier* o *clasificador*; y *attManager* o *gestor de atributos*. Cada agente tiene un artefacto *attManager* propio, pero sólo hay un *classifier artifact* con el que todos los agentes trabajan.

3.2.1. *AttManager Artifact* - Gestor de Atributos

Este artefacto encapsula objetos y métodos de Weka[13], tanto como para la evaluación de los atributos como la creación de los nodos. Es el que provee las operaciones necesarias para controlar la pila de particiones[1] (agregar y sacar elementos). Además por medio de conexiones con el artefacto *classifier*, envía a éste los resultados de las evaluaciones locales de los atributos y, en su caso, los nodos creados. Con el fin de preservar la privacidad de los datos, este artefacto no posee propiedades observables.

1. Operaciones.
   - *a*) **checkMultival**. Comprobar si alguno de los atributos es nominal y con muchos valores distintos.
   - *b*) **drop**. Retirar la última partición de la pila de particiones.
   - *c*) **evalAttributes**. Evaluar todos los atributos locales de la última partición en la pila de particiones.
   - *d*) **evalSplit**. Comprobar si todas las instancias pertenecen a una sola clase o si no hay suficientes instancias para partir.
   - *e*) **getTrainData**. Recibe los datos del artefacto *ExamplesBase*[2]. Crea la pila de particiones y coloca como primer elemento la base completa.

[1] La pila de particiones se explicará más a detalle en la sección de la implementación

[2] Artefacto originalmente propuesto en JaCa-DDM, se explicará su función en la sección de implementación, donde se describe la plataforma experimental.

f) **sendAttribute**/**sendLeaf**. Enviar un nuevo nodo de acuerdo con los resultados de la selección de atributos. Si hay una partición válida se construye el nodo con el atributo ganador, de lo contrario se construye un nodo hoja.

g) **sendMultival**. Enviar valor actual de la bandera-Multival al artefacto *classifier*.

h) **setMultival**. Actualizar bandera-Multival de acuerdo a los resultados de todos los participantes.

i) **sendValues**. Por medio de una conexión, enviar las evaluaciones de los atributos al artefacto *classifier*.

j) **setTraintmp**. Crea nuevas particiones y las añade al final de la pila.

3.2.2. *Classifier Artifact* - Clasificador

Encapsula objetos y métodos de Weka[13], para almacenar la estructura de árbol de decisión, además proporciona las operaciones para la coordinación entre agentes y la selección de atributos para realizar las particiones de acuerdo al algoritmo *J48*. Obtiene las evaluaciones de atributos y nodos nuevos para adjuntar.

1. Propiedad Observable.
   nodeStatus: muestra el estado del nodo actual.

   a) **0** Significa un nodo nuevo, por lo que los agentes tendrán que llevar a cabo un ciclo de aprendizaje.

   b) **1** El nodo actual tiene un hijo a procesar.

   c) **2** Nodo hoja o todos los hijos ya han sido procesados.

   d) **3** Cuando se realiza una acción de *BackTrack* y se alcanza la raíz.

2. Operaciones.

   a) **backtrack**. Mueve el apuntador del nodo actual al padre de éste, actualiza la propiedad observable *nodeStatus* y envía una señal de *backtrack* a todos los agentes.

   b) **checkValidM**. Verifica la bandera valid-models, es decir, si el proceso de evaluación de atributos dio como resultado al menos un modelo válido.

   c) **findBest**. Elige el mejor atributo y verifica si una partición útil fue encontrada.[3]

[3]De acuerdo a *J48*, una partición es útil, si el *gainRatio* comparado con cero, cumple con la menor desviación permitida, esto es 1e-6.

d) **getValues**. Para la estrategia *social*, permite al agente introducir los valores de las evaluaciones de los atributos dentro del artefacto.

e) **processAvInfG**. Calcula el *info-gain* promedio.

f) **sendRequest**. Envía una señal *requestatt* al agente con el atributo ganador.

g) **setReady**. Esta acción se emplea para coordinar el trabajo de los agentes, funciona como un checador, cuando todos los agentes "checan" en el artefacto, significa que están listos y se envía una señal para el agente *learner*.

3.3. Algoritmo de Aprendizaje

En la Figura 3.2 se muestra la interacción que se lleva a cabo entre agentes y artefactos a través del proceso de aprendizaje colaborativo.

A continuación una descripción más detallada del proceso:

1. El agente *learner*, invoca la operación *evalSplit* utilizando su artefacto *attManager* para verificar el primer criterio de paro: todas las instancias de la partición actual pertenecen a la misma clase; o no hay suficientes instancias para realizar una partición válida (mínimo dos instancias de acuerdo a J48).

    a) Si alguna de estas dos condiciones es cierta significará que es un nodo hoja. El agente *learner* mediante la operación *sendLeaf* de su artefacto *attManager*, crea el nodo correspondiente y lo transmite al artefacto *classifier*, el cual al recibir el nodo, emite una señal para indicarle a todos los agentes, que deben ejecutar la operación *drop* de sus respectivos artefactos *attManager*, para actualizar sus pilas de particiones.

2. De no cumplirse la condición de paro, se valida una bandera denominada *Multival*. De acuerdo a J48, ésta se vuelve falsa si al menos uno de los atributos es nominal y tiene muchos valores; el atributo puede pertenecer a cualquiera de los agentes. El proceso de validación se realiza de la siguiente manera:

    a) El agente *learner*, por medio de un acto de habla, le comunica a todos los participantes la intención de evaluar dicha bandera.

    b) Los agentes ejecutan la operación *checkMultival* empleando sus respectivos artefactos *attManager*.

    c) El resultado es compartido mediante la operación *sendMultival* de sus respectivos artefactos *attManager*, por medio de una conexión con el artefacto *classifier*.

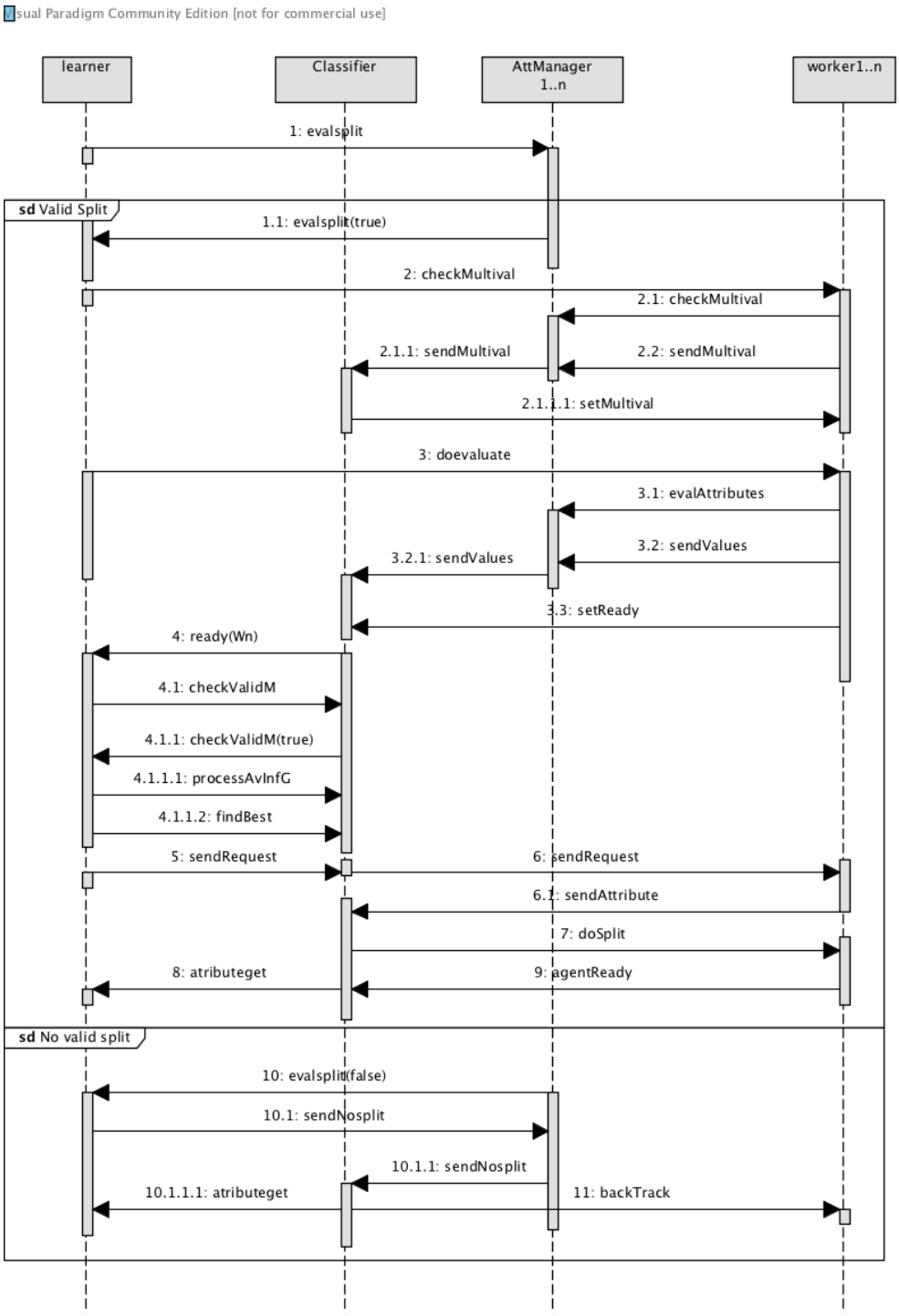

Figura 3.2: Proceso de Aprendizaje

d) El artefacto *classifier*, una vez que ha recibido todos los resultados de la validación, envía una señal que contiene el resultado global, es decir, falso si al menos un atributo cumplió la condición.

e) Los agentes al recibir la señal del artefacto *classifier*, actualizarán la bandera, de ser necesario, ejecutando la operación *setMultival* de sus artefactos *attManager*.

3. A continuación el agente *learner* solicita a todos los agentes evaluar sus atributos. Cada agente, evalúa localmente sus atributos ejecutando la operación *evalAttributes* con su artefacto *attManager*.

4. Hecho lo anterior, cada agente transmite una lista compuesta por: el nombre del agente, un identificador local para los atributos (el nombre del atributo no se emplea, se usa un número entero) y los resultados de la evaluación. Además de éstos se envía el número de modelos válidos encontrados. Finalmente cada agente notifica que está listo utilizando la operación *setReady* del artefacto *classifier*. [4]

5. Una vez que todos los agentes notifican estar listos, el artefacto *classifier* emite una señal al agente *learner*, el cual procede a evaluar el total de *modelos válidos* ejecutando la operación *checkValidM* de dicho artefacto. Hecho el cálculo, en caso de no encontrar modelos válidos, se actualiza uno de los criterios de paro del algoritmo y por lo tanto se ha encontrado un nodo hoja. El proceso de creación del nodo hoja, ya se explicó en el paso 1 inciso a).

6. Paso siguiente, el agente *learner* calcula la *ganancia de información promedio*, ejecutando la operación *processAvInfG* del artefacto *classifier*.

7. Después de los cálculos, el agente *learner*, utilizando el artefacto *classifier*, determina cuál es el mejor atributo para particionar los datos mediante la operación *findBest*. Dicha operación, valida que se encuentre una partición útil, de no encontrarse se efectúa el proceso para un nodo hoja.

8. Encontrado el mejor atributo, el artefacto *classifier* envía una señal con el identificador del atributo ganador al agente correspondiente.

[4] En esta etapa, se diseñaron e implementaron dos soluciones para poner a prueba, una denominada *social* y la otra *social mejorada*. En el enfoque *social*, los agentes después de evaluar de forma local sus atributos, le comunican directamente al agente *learner* los resultados por medio de *actos de habla*, éste introduce los datos al artefacto *classifier* y procede a la selección del mejor atributo. La versión *social mejorada* no permite la comunicación directa de los resultados, en su lugar éstos se comunican por medio de conexiones entre los artefactos, por lo que no se comparten directamente los datos entre los agentes, brindando una mayor privacidad a los datos.

9. El agente con el mejor atributo, utiliza la operación *sendAttribute* de su artefacto *attManager* para crear y enviar: el nodo correspondiente y, la o las particiones (una lista de enteros) que se derivan de elegir ese atributo al artefacto *classifier*.

10. Si el ciclo termina con la creación de un nodo hoja, se manda una señal de *backtrack* para los agentes participantes, que les indica que deben actualizar su pila de particiones para el siguiente ciclo, ejecutando la operación *drop*. Además de actualizar su pila de particiones, el agente *learner* ejecutará la operación *backtrack* del artefacto *classifier*, ésto actualiza la propiedad observable *nodeStatus* de dicho artefacto, si éste no tiene hijos pendientes de procesar, la operación *backtrack* se ejecutará de nuevo hasta alcanzar un nodo con hijos pendientes de procesar o la raíz, indicando el final del proceso.

3.4. Implementación

El agente *learner* controla el proceso de aprendizaje y, en ciertas etapas, solicita información a otros agentes, como por ejemplo, el valor de la *ganacia de información*; así que no es posible ejecutar el proceso de construcción del árbol de decisión de manera recursiva, porque cada agente realiza los cálculos de forma local.

Como se sabe, *J48* construye el árbol mediante una búsqueda primero en profundidad. La recursividad permite a cada nueva llamada obtener la partición de datos correspondiente que será procesada. Con el fin de replicar dicha funcionalidad, la estrategia de aprendizaje colaborativo propuesta, implementa una pila de particiones, donde las particiones son administradas. Los detalles particulares de dicha pila se explicarán en la siguiente sección dedicada a ésta.

Weka[13] fue hecho para el procesamiento centralizado, por lo que fue necesario realizar otra modificación. Internamente cuando se construye el árbol, sólo un identificador entero se almacena en la estructura, porque se espera que sólo habrá un esquema y los atributos no cambiarán de posición. Pero cuando esto se mueve al ambiente distribuido, cada parte puede tener un atributo identificado por el mismo número 1, 2, etc.

Para evitar referenciar incorrectamente los atributos, se modificó la estructura interna del árbol de decisión. Junto con el identificador, el nombre del atributo es almacenado internamente por el árbol, sin embargo, ningún agente puede verlo, porque sólamente se utiliza para el procesamiento interno que se ejecuta al clasificar un ejemplo, de esa manera, cuando se llama el proceso para clasificar, cada nodo referencía al atributo adecuado pero los nombres de los atributos no son revelados.

3.4.1. Pila de Particiones

Una pila (stack en inglés) es una lista ordenada o estructura de datos, en la que el modo de acceso a sus elementos es de tipo LIFO (del inglés Last In First Out, último en entrar, primero en salir) que permite almacenar y recuperar datos.

Esta forma de operar de la pila, es aprovechada para replicar el manejo de las particiones de datos que se lleva a cabo con la recursividad.

Para explicar cómo funciona, utilizaremos un sencillo ejemplo con una corrida del algoritmo de aprendizaje colaborativo propuesto. Supongamos que tenemos dos agentes, cada uno de ellos con dos de los atributos de la base de datos *tenis*, ver Tabla 3.3. El agente uno tendrá los atributos: outlook y temperature; y el agente dos: humidity y wind.

No.	outlook	temperature	humidity	wind	playTennis
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

Figura 3.3: Base de datos tenis, partición para dos agentes.

El agente uno actuará como *learner* y el dos como *worker*. El primer paso, es preparar sus datos para el proceso. Los agentes inicializarán sus pilas de particiones agregando el primer elemento, para el agente uno serán todos sus atributos: *outlook* y *temperature*; además de la clase, colocando los catorce ejemplos. El agente dos procede de la misma manera con sus atributos.

Como ya se explicó anteriormente, los agentes procederán a elegir el mejor atributo, es decir, el que ofrezca la mayor ganancia de información. Para nuestro ejemplo el ganador en esta etapa es el atributo *outlook* del agente uno. Elegir este atributo nos llevará a dividir la base en tres partes, de acuerdo a sus posibles valores que son: *sunny*, *overcast*, *rain*.

El agente uno, empleando su artefacto *attManager*, procede a crear el nodo correspondiente y anexarlo al árbol, cada vez que un nuevo nodo se crea, el apuntador interno del artefacto *classifier* que señala al nodo actual, se mueve para apuntar al nuevo nodo. Además de los pasos anteriores, el agente uno también crea la lista de listas con las particiones correspondientes, de acuerdo a los índices de las instancias y los valores del atributo seleccionado, quedando como sigue:

$$\begin{array}{c} \\ sunny \\ overcast \\ rain \end{array} \begin{bmatrix} (\ 1 & 2 & 8 & 9 & 11 &), \\ (\ 3 & 7 & 12 & 13 & &), \\ (\ 4 & 5 & 6 & 10 & 14 &) \end{bmatrix}$$

A continuación, los agentes proceden a crear y agregar las nuevas particiones de acuerdo a la lista referida. Tomando como base la última partición de sus respectivas pilas, cada agente creará tres nuevas particiones que se agregarán a la pila. En la Figura 3.4 podemos ver las pilas actualizadas de los agentes uno y dos.

No.	outlook	temperature	playTennis
1	sunny	hot	no
2	sunny	hot	No
8	sunny	mild	no
9	sunny	cool	yes
11	sunny	mild	yes

No.	outlook	temperature	playTennis
3	overcast	hot	yes
7	overcast	cool	yes
12	overcast	mild	yes
13	overcast	hot	yes

No.	outlook	temperature	playTennis
4	rain	mild	yes
5	rain	cool	yes
6	rain	cool	no
10	rain	mild	yes
14	rain	mild	no

No.	outlook	temperature	playTennis
1	sunny	hot	no
...	...	...	...
14	rain	mild	no

(a) Pila de particiones agente uno

No.	humidity	wind	playTennis
1	high	weak	no
2	high	strong	No
8	high	weak	no
9	normal	weak	yes
11	normal	strong	yes

No.	humidity	wind	playTennis
3	high	weak	yes
7	normal	strong	yes
12	high	strong	yes
13	normal	weak	yes

No.	humidity	wind	playTennis
4	high	weak	yes
5	normal	weak	yes
6	normal	strong	no
10	normal	weak	yes
14	high	strong	no

No.	humidity	wind	playTennis
1	high	weak	no
...	...	...	...
14	high	strong	no

(b) Pila de particiones agente dos

Figura 3.4: Pilas de particiones de los agentes uno y dos, después de una partición con el atributo *outlook*.

Como se observa en la figura, los agentes no requieren de mayor comunicación para coordinarse respecto al orden de las instancias, en el siguiente

ciclo si deciden evaluar los atributos para buscar una nueva partición, al tomar el último elemento de sus respectivas pilas, las instancias referenciadas son las mismas para ambos agentes.

En el siguiente ciclo, de acuerdo al protocolo propuesto, los agentes tomarán el último elemento de sus respectivas pilas de particiones, para proceder a evaluar si se hará una nueva partición. El agente uno evaluará *temperature*; mientras que el agente dos tomará en cuenta sus dos atributos: *humidity* y *wind*.

Cuando el agente uno solicite nuevamente que sean evaluados los atributos, el ganador será *humidity* del agente dos. El agente dos realizará la misma tarea que anteriormente llevó a cabo el agente uno. Crear el nodo correspondiente y anexarlo al árbol, y crear nuevamente las particiones respectivas, tomando como base a la última partición de su pila, y comunicar la lista correspondiente a los dos valores posibles de *humidity*: *high* y *normal*. La lista generada es la siguiente:

$$\begin{array}{c} high \\ normal \end{array} \left[\begin{array}{ccccc} (& 1 & 2 & 8 &), \\ (& 9 & 11 & &) \end{array} \right]$$

Una vez que las dos particiones derivadas de elegir el atributo *humidity* fueron creadas y anexadas, las pilas de particiones de ambos agentes tendrán ahora seis elementos. Actualizadas las pilas el ciclo se da por concluido dando paso a uno nuevo.

Para el siguiente ciclo al tomar la actual última partición de la pila, que contiene las instancias: 1, 2 y 8; se actualiza un criterio de paro, puesto que todas las instancias corresponden a la misma clase. El agente uno procederá a crear el nodo hoja correspondiente y anexarlo al árbol, hecho esto, por tratarse de una hoja, se emite una señal *backtrack* que le indica a los agentes que deben sacar el último elemento de sus pilas mediante la operación *drop*. Con la siguiente partición se llevará a cabo el mismo proceso ya que todas las instancias corresponden a la misma clase.

El proceso *backtrack*, como ya se mencionó con anterioridad, mueve el apuntador interno del artefacto *classifier* del nodo actual al padre de éste, por lo que en esta estapa, el nodo actual es aquel donde se eligió el atributo *humidity* del agente dos, ubicado en la rama del valor *sunny* del atributo *outlook*. La Figura 3.5 muestra el estado del árbol hasta la etapa actual, podemos observar que el nodo actual no tiene más hijos pendientes de procesar, internamente es visible por el agente a través de la propiedad observable *nodeStatus*.

Al encontrarse con un nodo sin hijos por procesar, el agente uno ejecutará la operación *backtrack*, por lo que el nodo actual cambiará a la raíz y ambos agentes ejecutarán la operación *drop*. En este momento las pilas de particiones ya sólo tienen tres elementos. La Figura 3.6 muestra el estado actual de las pilas de particiones.

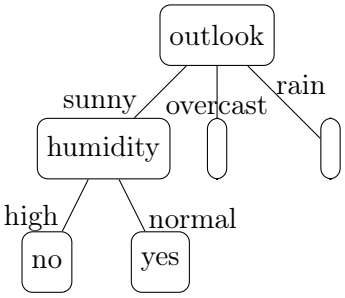

Figura 3.5: Árbol parcialmente construido para la base de datos tenis.

No.	outlook	temperature	playTennis
3	overcast	hot	yes
7	overcast	cool	yes
12	overcast	mild	yes
13	overcast	hot	yes

No.	outlook	temperature	playTennis
4	rain	mild	yes
5	rain	cool	yes
6	rain	cool	no
10	rain	mild	yes
14	rain	mild	no

No.	outlook	temperature	playTennis
1	sunny	hot	no
...	...	...	...
14	rain	mild	no

(a) Pila de particiones agente uno

No.	humidity	wind	playTennis
3	high	weak	yes
7	normal	strong	yes
12	high	strong	yes
13	normal	weak	yes

No.	humidity	wind	playTennis
4	high	weak	yes
5	normal	weak	yes
6	normal	strong	no
10	normal	weak	yes
14	high	strong	no

No.	humidity	wind	playTennis
1	high	weak	no
...	...	...	...
14	high	strong	no

(b) Pila de particiones agente dos

Figura 3.6: Pilas de particiones de los agentes uno y dos, después cinco ciclos de ejecución.

El siguiente ciclo produce una nueva hoja al árbol, pues si observamos la última partición de la pila del agente uno, todos los ejemplos corresponden a la misma clase. El proceso será el mismo que ya se ha explicado, dando lugar a un nuevo nodo y provocando que los agentes actualicen sus pilas nuevamente, quedando con sólo dos elementos en las mismas.

A continuación los agentes procesarán la partición con los indices: 4, 5, 6, 10 y 14. Cuando el agente uno solicite que los atributos sean evaluados, el proceso se realizará de forma muy similar a lo hecho al principio cuando se eligió el atributo *humidity* del agente dos. De acuerdo a la partición que será procesada, el atributo ganador es *wind* del agente dos, el cual sólo cuenta con dos valores posibles: *weak* y *strong*. Esto nos llevará a crear dos nuevas

particiones:

$$\begin{array}{c} \textit{weak} \\ \textit{strong} \end{array} \left[\begin{array}{ccccc} (& 4 & 5 & 10 &), \\ (& 6 & 14 & &) \end{array} \right]$$

Ambas particiones dividen los ejemplos de tal manera, que todas las instancias, en cada partición respectivamente, pertenecen a una sola clase, lo que nos llevará a nodos hoja, realizando un proceso como el que ya fue explicado con anterioridad. Para esta etapa el árbol ha sido construido por completo, en la Figura 3.7 podemos observar el árbol resultante.

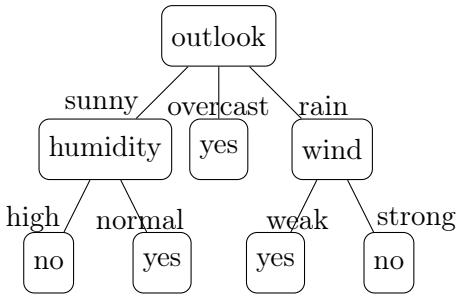

Figura 3.7: Árbol constriudo para la base de datos tenis.

Llegado a esta etapa, el proceso de *backtrack* nos llevará a vaciar las pilas de particiones y moverá el apuntador a la raíz. Así con la pila vacía y sin más nodos que procesar finaliza el proceso de aprendizaje.

3.4.2. Plataforma Experimental

JaCa-DDM [15] proporciona una herramienta para configurar los experimentos, crea particiones de datos estratificados aleatorios y los distribuye entre los agentes, pero está diseñado para datos horizontalmente particionados. Este trabajo de cargar el conjunto de datos y hacer todas las particiones para el experimento se lleva a cabo por el artefacto *oracle*. Éste carga un archivo ARFF y crea particiones aleatorias de acuerdo con el número de agentes que tendrán lugar en el proceso de aprendizaje. El artefacto *oracle* original fue modificado para que pueda crear datos con particionamiento vertical.

Para el diseño experimental, se desarrolló un sistema basado en agentes cuya función es preparar el ambiente necesario para efectuar los experimentos. Esto debido a que en este trabajo las pruebas se realizaron de forma concurrente, por lo que se tomó la base de datos a probar y se particionó de manera que pueda emular la distribución vertical de los datos, es decir, los atributos se distribuyeron de forma aleatoria de acuerdo al número de agentes participantes.

El proceso se compone de los siguientes pasos:

- El agente *organizer*, mediante el artefacto *oracle*, carga la base de datos contenida en un archivo con extensión arff, cuyo formato está estandarizado para la herramienta *Weka*[13].
- Una vez cargada la base, empleando el mismo artefacto genera las particiones *train* y *test*, la primera para crear el modelo y la segunda para realizar las pruebas.
- En cuanto a la partición *train*, es dividida en partes por cada uno de los agentes que participarán en el proceso de aprendizaje.
- Primero se genera una lista aleatoria (lista de números enteros) que representan cada uno de los atributos, a continuación se recorre la lista mediante un proceso de ruleta, asignando uno por uno los atributos a cada una de las n particiones correspondientes a los n agentes, hasta que la lista se vacía. Esta lista se guarda por lo que todas las estrategias, la centralizada y sociales, obtienen el mismo esquema de cada ciclo, para el método de validación cruzada cada *fold* obtiene las mismas particiones de atributos, sólo las instancias en cada partición cambian.
- Listas las particiones, el agente *organizer* notifica a los agentes que participarán en el aprendizaje, para que cada uno, obtenga su partición del artefacto *oracle*, mediante una conexión entre el artefacto *examples_ base* y el artefacto *oracle*.

El artefacto *examples_ base* original de JaCa-DDM, se encarga de almacenar y gestionar los ejemplos del agente. Proporciona las operaciones necesarias para el proceso de aprendizaje en esa herremienta, como son:

- Realizar una solicitud al artefacto *Oracle*, para que éste envíe los ejemplos de entrenamiento correspondientes.
- Enviar todo los ejemplos al artefacto *ClassifierJ48*. Pues en JaCa-DDM, el coordinador debe enviar todos sus ejemplos de entrenamiento para crear el modelo base.
- Obtener el modelo de entrenamiento actual, a fin de realizar una búsqueda de contradicciones en la base de ejemplos y, en caso de encontrarla, enviar el ejemplo contradictorio al artefacto *ClassifierJ48*.

El artefacto *examples_ base* original, se reutilizó a fin de que sea el encargado de obtener los ejemplos de entrenamiento del artefacto *Oracle*. Pero dado que en esta propuesta, las operaciones propias del aprendizaje, están divididas entre las que se realizan de forma local y aquéllas que deben llevarse a cabo de forma distribuida; se creó el artefacto *attManager* para las operaciones locales y el artefacto *classifier* se encarga de las operaciones distribuidas.

Capítulo 4

Resultados y Discusión

4.1. Diseño Experimental

La Tabla 4.1 contiene las bases de datos utilizadas para los experimentos reportados, todas obtenidas del repositorio *UCI machine learning repository*[2]. Se elegió las bases de datos buscando las siguientes características:

- Bases de datos con un número elevado de atributos y un número relativamente menor de instancias. Bases *australian* y *heart*;
- Con un número de atributos relativamente menor y un mayor número de instancias. Bases *breast* y *car*;
- La base de datos *german* se eligió por tener relativamente, un número mayor de instancias y atributos.
- Las bases *letter*, *mushroom* y *waveform*; representan el escenario con un número alto de instancias y atributos.

Tabla 4.1: Bases de datos utilizadas para los experimentos.

Base de Datos	**Instancias**	**Atributos**	**Valores de Clase**
australian	690	14	2
breast	683	9	2
car	1728	6	4
german	1000	20	2
heart	270	13	2
iris	150	4	3
letter	20000	17	26
mushroom	8124	22	2
waveform	5000	40	3

- La base de datos *iris*, se eligió por ser ampliamente conocida y utilizada debido a la precisión con que fueron obtenidos sus datos.

Se aplicó la técnica de *cross-validation*[14] o validación cruzada con estratificación de los datos, empleando un valor de *fold* igual a diez, llevando a cabo 10 repeticiones por cada base de datos. Para cada base de datos, los experimentos se realizaron con combinaciones de 2, 4, 7, 10, 15 y 20 agentes. Dado que el número de atributos es limitado, las combinaciones se probaron sólo en el caso de ser suficientes éstos para repartir por lo menos uno a cada agente.

Para comparar si existen diferencias significativas entre las tres estrategias respecto a la certeza de clasificación, se utilizó la prueba T pareada de dos colas con 0.05 grados de significancia. Específicamente se utilizó la versión denominada *corrected resampled t-test statistic*[17] implementada dentro de Weka[13] para la prueba estadística.

Esta prueba recibe dos vectores que se componen de todos los resultados en la certeza de clasificación obtenidos por las dos estrategias a comparar, diez por cada ciclo de *cross-validation* por diez repeticiones, en total cien. La función devuelve un valor entero de acuerdo a lo siguiente: 0 significa que no existen diferencias significativas entre ambos modelos; 1 significa que la primera estrategia presenta resultados mejores estadísticamente significativos con respectoa a la segunda estrategia; y -1 significa que la primera estrategia presenta resultados peores estadísticamente significativos con respectoa a la segunda estrategia.

4.2. Resultados

La Tabla 4.2 reporta las mediciones de certeza en la clasificación, obtenidas por cada modelo construido. Se muestra la media de la certeza correspondiente para cada conjunto de datos y número de agentes probado.

La Tabla 4.3 reporta las mediciones del tiempo requerido para la creación del modelo. Se muestra la media de los tiempos obtenidos correspondiente a cada conjunto de datos y número de agentes probado, medido en milisegundos.

En las Tablas 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10 y 4.11 se muestran los estadísticos correspondientes a las ejecuciones del conjunto de bases de datos probadas, de acuerdo al número de agentes participantes, en cuanto a certeza en la clasificación.

En las Tablas 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.18, 4.19 y 4.20 se muestran los estadísticos correspondientes a las ejecuciones del conjunto de bases de datos probadas, de acuerdo al número de agentes participantes, en cuanto a tiempo necesario para la construcción del modelo, medido en milisegundos.

Tabla 4.2: Certeza de clasificación. La primera columna indica la base de datos de prueba, la segunda el número de agentes que participaron en el proceso de aprendizaje, las siguientes tres columnas presentan la media en la certeza de clasificación obtenida por los modelos: centralizado, social y social mejorado respectivamente. Las últimas tres columnas contienen el resultado de la prueba T pareada: la primera columna corresponde a la comparación entre el modelo centralizado tradicional y el modelo social, la segunda es el modelo centralizado contra la mejora del modelo social y la tercera es el modelo social contra la mejora del modelo social.

Base	**A.**	**Cent.**	**Social**	**SocialM**	**CvsS**	**CvsSM**	**SvsSM**
australian	2	83.6377	83.6667	83.7101	0	0	0
australian	4	83.6377	83.5797	83.6377	0	0	0
australian	7	83.6232	83.5652	83.7681	0	0	0
australian	10	83.7246	83.6812	83.6667	0	0	0
breast	2	95.6215	95.7089	95.6503	0	0	0
breast	4	96.0934	96.0786	96.0782	0	0	0
breast	7	96.0735	96.0586	96.1172	0	0	0
car	2	93.4785	93.3685	93.3743	0	0	0
car	4	93.7211	93.6343	93.6575	0	0	0
german	2	68.2600	68.1400	68.1900	0	0	0
german	4	68.8100	68.6100	68.6700	0	0	0
german	7	68.4300	68.4700	68.5500	0	0	0
german	10	69.1600	69.1300	69.1500	0	0	0
german	15	68.6300	68.6100	68.6200	0	0	0
german	20	68.8000	68.9000	68.7000	0	0	0
heart	2	77.1111	77.1111	77.0741	0	0	0
heart	4	77.4444	77.5185	77.6667	0	0	0
heart	7	77.2963	77.5926	77.4074	0	0	0
heart	10	77.3333	76.9259	77.3333	0	0	0
iris	2	93.8000	93.7333	93.6000	0	0	0
iris	4	94.8667	94.8000	94.8000	0	0	0
letter	2	88.0065	88.0295	88.0210	0	0	0
letter	4	88.0775	88.0835	88.0575	0	0	0
letter	7	88.1155	88.1200	88.1025	0	0	0
letter	10	87.9380	87.9295	87.9520	0	0	0
letter	15	87.9825	87.9825	88.0105	0	0	0
mushroom	2	100.0000	100.0000	100.0000	0	0	0
mushroom	4	100.0000	100.0000	100.0000	0	0	0
mushroom	7	100.0000	100.0000	100.0000	0	0	0
mushroom	10	100.0000	100.0000	100.0000	0	0	0
mushroom	15	100.0000	100.0000	100.0000	0	0	0
mushroom	20	100.0000	100.0000	100.0000	0	0	0
waveform	2	75.2400	75.3220	75.3720	0	0	0
waveform	4	75.2040	75.1500	75.3240	0	0	0
waveform	7	75.5400	75.6520	75.5960	0	0	0
waveform	10	75.3740	75.4080	75.4460	0	0	0
waveform	15	75.3460	75.5020	75.5160	0	0	0
waveform	20	75.5900	75.4300	75.4940	0	0	0

Tabla 4.3: Tiempo de creación del modelo, medido en milisegundos. La primera columna indica la base de datos de prueba, la segunda el número de agentes que participaron en el proceso de aprendizaje, las siguientes tres columnas presentan la media del tiempo para crear el modelo: centralizado, social y social mejorado respectivamente. Las últimas tres columnas contienen el resultado de la prueba T pareada: la primera corresponde a la comparación entre el modelo centralizado tradicional y el modelo social, la segunda es el modelo centralizado contra la mejora del modelo social y la tercera es el modelo social contra la mejora del modelo social.

Base	**A.**	**Cent.**	**Social**	**SocialM**	**CvsS**	**CvsSM**	**SvsSM**
australian	2	8.15	6,848.25	5,248.68	-1	-1	0
australian	4	7.75	6,880.79	5,483.51	-1	-1	0
australian	7	7.85	8,881.66	9,328.11	-1	-1	0
australian	10	7.89	7,339.87	7,490.95	-1	-1	0
breast	2	3.85	2,251.65	2,186.84	-1	-1	0
breast	4	4.08	3,018.82	2,595.51	-1	-1	0
breast	7	4.55	4,704.22	4,558.54	-1	-1	0
car	2	6.26	6,608.78	5,434.17	-1	-1	0
car	4	6.43	8,611.05	7,747.26	-1	-1	0
german	2	12.75	12,843.72	7,140.47	-1	-1	1
german	4	13.02	11,278.33	9,318.01	-1	-1	0
german	7	13.98	32,178.03	32,221.80	-1	-1	0
german	10	14.15	24,781.82	25,154.03	-1	-1	0
german	15	15.27	59,475.05	57,788.06	-1	-1	0
german	20	15.61	124,261.13	128,091.83	-1	-1	0
heart	2	3.99	3,080.62	1,702.95	-1	-1	1
heart	4	4.41	4,148.98	3,676.86	-1	-1	0
heart	7	4.59	5,269.74	5,508.55	-1	-1	0
heart	10	4.66	4,188.10	3,797.46	-1	-1	0
iris	2	3.48	775.19	698.34	-1	-1	0
iris	4	3.79	711.56	771.53	-1	-1	0
letter	2	3,386.61	97,067.45	80,622.19	-1	-1	1
letter	4	3,454.92	151,577.55	122,904.79	-1	-1	1
letter	7	3,441.29	196,337.16	178,184.22	-1	-1	1
letter	10	3,373.14	181,797.80	170,953.10	-1	-1	1
letter	15	3,454.74	398,810.68	352,934.27	-1	-1	1
mushroom	2	30.07	1,302.95	1,124.56	-1	-1	0
mushroom	4	31.82	1,021.48	957.93	-1	-1	0
mushroom	7	32.85	1,838.37	1,877.59	-1	-1	0
mushroom	10	34.29	2,229.76	2,245.37	-1	-1	0
mushroom	15	36.21	3,703.65	2,040.21	-1	-1	1
mushroom	20	36.86	7,814.06	5,365.93	-1	-1	1
waveform	2	808.59	20,076.27	15,279.44	-1	-1	1
waveform	4	817.30	23,973.18	18,600.12	-1	-1	1
waveform	7	821.73	33,587.63	31,784.22	-1	-1	0
waveform	10	825.37	26,397.31	21,177.41	-1	-1	1
waveform	15	828.80	33,652.16	22,976.08	-1	-1	1
waveform	20	833.82	105,877.91	95,614.76	-1	-1	1

Tabla 4.4: Estadísticas de las pruebas realizadas con la base de datos australian, en cuanto a certeza en la clasificación.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	85.072	84.638	84.928
	Mediana	83.333	83.623	83.768
	Peor	82.899	82.754	82.899
	σ	0.728	0.641	0.694
4	Mejor	84.638	84.348	84.493
	Mediana	83.768	83.768	83.768
	Peor	82.174	82.174	82.174
	σ	0.646	0.707	0.678
7	Mejor	85.652	85.507	85.652
	Mediana	83.478	83.333	83.623
	Peor	82.609	82.609	82.899
	σ	0.848	0.894	0.794
10	Mejor	84.783	84.638	84.638
	Mediana	84.058	83.913	83.768
	Peor	82.899	82.609	82.754
	σ	0.773	0.649	0.720

Tabla 4.5: Estadísticas de las pruebas realizadas con la base de datos breast, en cuanto a certeza en la clasificación.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	96.481	96.628	96.628
	Mediana	95.750	95.612	95.750
	Peor	94.872	94.725	94.725
	σ	0.528	0.599	0.548
4	Mejor	96.496	96.496	96.641
	Mediana	96.194	96.338	96.189
	Peor	95.173	94.879	95.465
	σ	0.404	0.488	0.324
7	Mejor	96.630	96.775	96.775
	Mediana	96.200	96.200	96.200
	Peor	95.307	95.452	95.452
	σ	0.447	0.457	0.445

Tabla 4.6: Estadísticas de las pruebas realizadas con la base de datos car, en cuanto a certeza en la clasificación.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	93.865	93.809	93.809
	Mediana	93.461	93.345	93.287
	Peor	93.114	92.998	92.998
	σ	0.276	0.261	0.276
4	Mejor	94.096	94.039	94.154
	Mediana	93.807	93.634	93.691
	Peor	93.287	93.286	93.229
	σ	0.233	0.215	0.250

Tabla 4.7: Estadísticas de las pruebas realizadas con la base de datos german, en cuanto a certeza en la clasificación.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	69.400	69.700	69.700
	Mediana	68.300	67.900	68.000
	Peor	67.400	66.900	67.200
	σ	0.763	0.922	0.827
4	Mejor	70.100	69.800	69.500
	Mediana	69.000	69.000	69.300
	Peor	67.700	67.000	67.200
	σ	0.782	1.054	0.879
7	Mejor	69.800	69.600	69.400
	Mediana	68.900	68.800	69.100
	Peor	66.300	66.500	66.900
	σ	1.064	0.936	0.822
10	Mejor	71.600	71.600	71.700
	Mediana	69.300	69.000	69.100
	Peor	67.300	67.100	67.100
	σ	1.297	1.418	1.415
15	Mejor	69.400	69.600	69.600
	Mediana	68.800	68.700	68.800
	Peor	67.700	67.700	67.700
	σ	0.652	0.559	0.581
20	Mejor	70.300	70.400	70.300
	Mediana	68.600	69.100	68.900
	Peor	67.700	67.700	67.600
	σ	0.909	0.948	0.903

Tabla 4.8: Estadísticas de las pruebas realizadas con la base de datos heart, en cuanto a certeza en la clasificación.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	78.519	78.148	78.148
	Mediana	77.407	77.037	77.037
	Peor	76.296	75.926	75.926
	σ	0.834	0.757	0.915
4	Mejor	79.259	79.259	79.259
	Mediana	77.407	77.778	78.148
	Peor	75.926	75.556	75.926
	σ	1.068	1.185	1.272
7	Mejor	80.000	79.630	80.000
	Mediana	77.407	77.778	77.407
	Peor	75.556	75.926	75.926
	σ	1.247	1.135	1.132
10	Mejor	79.259	79.630	80.000
	Mediana	77.037	77.037	77.037
	Peor	75.926	74.815	74.444
	σ	1.416	1.610	1.779

Tabla 4.9: Estadísticas de las pruebas realizadas con la base de datos iris, en cuanto a certeza en la clasificación.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	95.333	95.333	95.333
	Mediana	94.000	94.000	94.000
	Peor	92.000	92.000	92.000
	σ	1.135	0.953	1.098
4	Mejor	96.000	96.000	96.000
	Mediana	94.667	94.667	94.667
	Peor	94.000	93.333	93.333
	σ	0.632	0.757	0.757

Tabla 4.10: Estadísticas de las pruebas realizadas con la base de datos letter, en cuanto a certeza en la clasificación.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	88.190	88.250	88.250
	Mediana	88.035	88.055	88.085
	Peor	87.650	87.715	87.670
	σ	0.154	0.182	0.171
4	Mejor	88.380	88.380	88.280
	Mediana	88.165	88.145	88.110
	Peor	87.675	87.700	87.690
	σ	0.197	0.181	0.168
7	Mejor	88.280	88.225	88.275
	Mediana	88.115	88.145	88.105
	Peor	87.945	87.990	87.975
	σ	0.095	0.078	0.087
10	Mejor	88.165	88.180	88.180
	Mediana	87.990	88.005	87.995
	Peor	87.540	87.475	87.555
	σ	0.193	0.199	0.187
15	Mejor	88.295	88.365	88.325
	Mediana	88.015	88.005	87.970
	Peor	87.800	87.745	87.820
	σ	0.179	0.195	0.160

Tabla 4.11: Estadísticas de las pruebas realizadas con la base de datos waveform, en cuanto a certeza en la clasificación.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	75.680	75.820	76.160
	Mediana	75.380	75.320	75.340
	Peor	74.740	74.620	74.780
	σ	0.375	0.395	0.402
4	Mejor	75.800	75.540	75.780
	Mediana	75.140	75.160	75.340
	Peor	74.720	74.500	74.740
	σ	0.371	0.295	0.262
7	Mejor	76.340	76.400	76.400
	Mediana	75.460	75.820	75.660
	Peor	74.660	74.760	74.760
	σ	0.504	0.529	0.593
10	Mejor	75.860	76.000	76.000
	Mediana	75.540	75.560	75.560
	Peor	74.560	74.280	74.540
	σ	0.451	0.525	0.466
15	Mejor	75.820	75.760	75.900
	Mediana	75.480	75.640	75.660
	Peor	74.260	74.960	75.020
	σ	0.466	0.266	0.304
20	Mejor	76.380	76.220	76.180
	Mediana	75.480	75.460	75.500
	Peor	74.920	74.260	74.960
	σ	0.453	0.598	0.436

Tabla 4.12: Estadísticas de las pruebas realizadas con la base de datos australian, respecto del tiempo necesario para la construcción del modelo, medido en milisegundos.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	6.900	6075.300	4184.400
	Mediana	7.300	6823.300	5145.300
	Peor	14.200	7632.400	6310.000
	σ	2.176	452.713	710.264
4	Mejor	6.600	5291.200	4519.500
	Mediana	7.300	6599.200	5529.100
	Peor	12.300	8080.500	6341.600
	σ	1.641	895.065	531.646
7	Mejor	6.400	7671.600	8037.900
	Mediana	7.400	8850.700	9044.200
	Peor	12.800	10174.200	10576.200
	σ	1.796	753.311	788.057
10	Mejor	6.800	6212.800	6020.100
	Mediana	7.200	7147.500	7603.000
	Peor	13.200	8436.200	8455.400
	σ	1.916	678.663	827.692

Tabla 4.13: Estadísticas de las pruebas realizadas con la base de datos breast, respecto del tiempo necesario para la construcción del modelo, medido en milisegundos.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	2.900	1689.600	1527.800
	Mediana	3.300	2023.800	2282.200
	Peor	8.200	2890.600	2593.200
	σ	1.608	440.431	362.006
4	Mejor	3.000	1532.600	1530.300
	Mediana	3.300	3249.600	2502.500
	Peor	8.300	3969.900	3214.600
	σ	1.597	796.545	534.189
7	Mejor	3.600	3681.400	3171.000
	Mediana	3.900	4586.700	4650.200
	Peor	8.200	5852.500	5607.800
	σ	1.492	667.690	825.440

Tabla 4.14: Estadísticas de las pruebas realizadas con la base de datos car, respecto del tiempo necesario para la construcción del modelo, medido en milisegundos.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	4.900	4627.900	3367.500
	Mediana	5.400	6944.500	4949.200
	Peor	12.500	8359.000	7941.100
	σ	2.267	1459.181	1564.850
4	Mejor	5.100	6358.100	5802.100
	Mediana	5.700	8721.500	7726.000
	Peor	12.000	9528.100	9208.900
	σ	2.054	957.528	1103.520

Tabla 4.15: Estadísticas de las pruebas realizadas con la base de datos german, respecto del tiempo necesario para la construcción del modelo, medido en milisegundos.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	11.400	11080.400	4678.500
	Mediana	12.000	11874.200	6457.100
	Peor	19.100	17395.200	13008.300
	σ	2.276	2036.307	2246.186
4	Mejor	11.300	10644.000	8433.900
	Mediana	12.500	11365.200	8921.400
	Peor	18.000	11838.000	11422.400
	σ	1.815	418.851	883.310
7	Mejor	13.000	30767.400	30650.800
	Mediana	13.500	31987.600	31831.500
	Peor	18.400	33293.000	34405.300
	σ	1.599	853.668	1283.734
10	Mejor	13.000	23105.100	22969.200
	Mediana	13.600	24965.900	25068.500
	Peor	18.800	27083.700	27030.300
	σ	1.719	1170.181	1224.945
15	Mejor	14.100	57698.700	54736.100
	Mediana	14.800	59305.300	57958.700
	Peor	19.800	61880.200	61286.000
	σ	1.651	1456.906	1873.128
20	Mejor	14.600	119128.200	125081.800
	Mediana	15.100	122723.700	128192.000
	Peor	18.900	132919.100	131449.000
	σ	1.242	3814.876	2060.169

Tabla 4.16: Estadísticas de las pruebas realizadas con la base de datos heart, respecto del tiempo necesario para la construcción del modelo, medido en milisegundos.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	3.000	2557.100	1091.000
	Mediana	3.400	3107.900	1393.900
	Peor	9.100	3628.900	2420.500
	σ	1.828	354.606	539.923
4	Mejor	3.500	3502.300	2686.600
	Mediana	3.900	4140.500	3695.500
	Peor	9.500	4837.400	4453.600
	σ	1.805	427.791	622.271
7	Mejor	3.900	4612.200	4536.600
	Mediana	4.000	5165.800	5400.800
	Peor	8.500	5824.900	6510.100
	σ	1.398	400.723	659.317
10	Mejor	3.900	3325.300	3000.500
	Mediana	4.000	4223.900	3767.200
	Peor	9.100	4799.200	4520.600
	σ	1.587	450.047	384.184

Tabla 4.17: Estadísticas de las pruebas realizadas con la base de datos iris, respecto del tiempo necesario para la construcción del modelo, medido en milisegundos.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	2.300	678.200	567.500
	Mediana	3.000	729.200	668.900
	Peor	7.300	1196.800	1149.200
	σ	1.533	164.301	170.534
4	Mejor	2.700	580.300	571.300
	Mediana	3.100	681.700	722.700
	Peor	8.100	964.600	1302.200
	σ	1.593	114.761	208.958

Tabla 4.18: Estadísticas de las pruebas realizadas con la base de datos letter, respecto del tiempo necesario para la construcción del modelo, medido en milisegundos.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	3294.800	85442.100	76900.100
	Mediana	3368.600	99404.600	80311.800
	Peor	3454.800	101951.200	83508.000
	σ	51.960	5159.910	2208.934
4	Mejor	3331.300	137863.700	108560.600
	Mediana	3449.800	154512.500	123361.500
	Peor	3561.600	161791.700	134444.000
	σ	76.119	9347.770	8638.284
7	Mejor	3293.900	163966.500	155433.200
	Mediana	3369.900	168690.800	157529.200
	Peor	3756.200	231791.300	214864.700
	σ	135.887	31735.469	25232.958
10	Mejor	3289.200	163281.100	156067.600
	Mediana	3346.000	167865.000	159967.700
	Peor	3629.500	218439.400	202954.400
	σ	97.108	23277.707	20050.210
15	Mejor	3232.200	338693.800	282092.700
	Mediana	3447.000	414513.800	369634.000
	Peor	3839.700	423787.000	384539.600
	σ	156.772	34510.197	38375.115

Tabla 4.19: Estadísticas de las pruebas realizadas con la base de datos mushroom, respecto del tiempo necesario para la construcción del modelo, medido en milisegundos.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	26.900	1090.600	753.100
	Mediana	29.100	1297.300	1009.600
	Peor	34.800	1810.600	1650.000
	σ	2.381	216.950	316.334
4	Mejor	29.600	699.100	824.100
	Mediana	31.600	1056.700	972.900
	Peor	34.300	1198.700	1131.700
	σ	1.168	154.536	108.311
7	Mejor	30.200	1443.200	1404.600
	Mediana	32.200	1747.600	1860.400
	Peor	36.100	2348.400	2590.800
	σ	1.665	294.816	382.945
10	Mejor	33.000	1866.400	1788.300
	Mediana	34.000	2094.300	2302.700
	Peor	37.700	2830.100	2696.600
	σ	1.374	316.501	300.390
15	Mejor	34.900	3083.700	1579.100
	Mediana	35.300	3770.100	2091.900
	Peor	40.500	4178.000	2292.900
	σ	1.845	345.727	217.681
20	Mejor	34.100	5421.500	3106.000
	Mediana	35.200	7885.400	5549.200
	Peor	42.000	8521.000	6419.100
	σ	2.614	883.959	883.202

Tabla 4.20: Estadísticas de las pruebas realizadas con la base de datos waveform, respecto del tiempo necesario para la construcción del modelo, medido en milisegundos.

Agentes		**Centralizada**	**Social**	**Social Mejorada**
2	Mejor	801.200	19591.200	14663.300
	Mediana	808.400	19823.900	15076.100
	Peor	817.000	21372.300	16934.300
	σ	5.662	591.933	661.361
4	Mejor	811.000	20970.600	16993.200
	Mediana	819.000	23799.900	18379.900
	Peor	826.100	25589.800	21120.800
	σ	5.363	1370.396	1219.410
7	Mejor	811.900	32488.600	29536.800
	Mediana	822.500	33334.000	31536.800
	Peor	828.700	35688.300	33271.200
	σ	5.302	1137.491	1331.160
10	Mejor	816.100	24304.200	20042.500
	Mediana	826.200	26425.000	20984.600
	Peor	836.100	27924.600	22333.600
	σ	6.523	1069.164	659.423
15	Mejor	816.900	31889.600	21095.000
	Mediana	830.000	33230.300	22711.400
	Peor	835.700	37141.100	25243.100
	σ	5.543	1608.654	1606.185
20	Mejor	818.700	101113.000	93153.700
	Mediana	834.900	105937.600	95073.400
	Peor	845.800	110541.300	101393.900
	σ	8.542	3159.983	2328.785

4.3. Discusión

En cuanto a la certeza en la clasificación, el desempeño de las estrategias sociales es equivalente al de la estrategia centralizada tradicional, no se encontraron diferencias significativas entre los modelos obtenidos por las estrategias sociales y aquéllos generados con la estrategia centralizada. El particionamiento vertical de los datos, no fue un impedimento para obtener un modelo equivalente al obtenido con el enfoque centralizado.

Respecto al número de agentes que participa en el proceso de aprendizaje, particularmente para las estrategias sociales, éste no impacta en la calidad del modelo obtenido. De los resultados se puede observar que independientemente del número de agentes que participaron en el proceso de aprendizaje, los valores de certeza en la clasificación obtenidos no presentan variaciones importantes, lo cual se corrobora con bajos valores en la desviación estándar.

Para la estrategia centralizada, en cuanto a la construcción del modelo, el número de agentes participantes es irrelevante, pues el proceso de aprendizaje se lleva a cabo por un solo agente siempre.

Caso distinto se presenta en las mediciones de tiempo, por ejemplo, para la estrategia centralizada, los tiempos aumentan conforme el número de agentes que participa en el proceso de aprendizaje aumenta. Sin embargo, los agentes en esta estrategia, colaboran únicamente para centralizar los datos, mientras que el aprendizaje es realizado por sólo un agente, por ello el impacto en el tiempo es bajo, si observamos por ejemplo los resultados mostrados en la Tabla 4.20, la media del tiempo para la estrategia centralizada, cuando participan veinte agentes, es únicamente un tres por ciento mayor que cuando participan dos agentes.

Igualmente ocurre para las estrategias sociales, donde a mayor número de participantes, mayor tiempo requerido para la construción del modelo. Pero en el caso de éstas, el impacto del número de agentes es mayor, pues los agentes participan durante todo el proceso de aprendizaje. Otro factor a tomar en cuenta, es la asignación de recursos por parte del sistema operativo, el lenguje de programación orientado a agentes *Jason*, no permite establecer que, por ejemplo, cada agente sea asignado a un procesador diferente, por lo que si bien se considera que los agentes son ejecutados en paralelo, la asignación real de tiempo de procesamiento corre a cargo del sistema operativo, es decir la ejecución en paralelo no está garantizada.

El enfoque centralizado tradicional necesita un menor tiempo para la construción del modelo respecto de las estartegias sociales. Sin embargo esto se esperaba, puesto que para cada ciclo del algoritmo de aprendizaje, en este caso *J48*, el enfoque centralizado sólo consume tiempo en la evaluación de los atributos y el proceso de selección del mejor de éstos; en cambio para los enfoques sociales, se necesita, además de realizar dichos pasos, en el mejor de los casos tiempo para procesar el ciclo más largo de razonamiento de los

agentes (para el caso paralelo o concurrente), pero en el peor de los casos, el tiempo de razonamiento será la suma de los tiempos de todos los agentes (no concurrente).

Los sistemas multi-agente, a diferencia de un algoritmo como *J48*, no se ejecutan de manera secuencial. En el caso de este trabajo, la estrategia centralizada tradicional ejecuta los pasos del algoritmo de aprendizaje de manera secuencial, por ello consume un menor tiempo; mientras que las estrategias sociales, ejecutan de manera secuencial únicamente algunos procesos/funciones del algoritmo, como por ejemplo: el cálculo de la ganancia de información o evaluar una bandera; entre otros. En la estrategia social, el agente *learner* no ejecuta sus acciones de manera secuencial, cada ciclo de razonamiento debe estar atento a diferentes aspectos como las respuestas de los demás agentes o las señales que emita el artefacto *classifier*; además de las acciones que se deriven de las intenciones previamente adoptadas.

De los resultados se puede concluir que el enfoque social mejorado, para las bases de datos más grandes, significativamente emplea un tiempo menor para la generación del modelo que el enfoque social, al mismo tiempo que la certeza en la clasificación se mantiene. Por lo anterior, se puede afirmar que la comunicación entre los artefactos es más eficiente, para el caso del envío de los datos estadísticos, que la llevada a cabo entre los agentes, ya que éstos últimos deben atender a su ciclo de razonamiento a la vez que realizan la comunicación.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

De acuerdo a los resultados obtenidos y descritos en el capítulo anterior, podemos concluir que la hipótesis propuesta en este trabajo es validada, la implementación basada en el paradigma de agentes y artefactos, obtiene niveles competitivos en cuanto a certeza en la clasificación, sin que sea necesario compartir directamente los datos. Sin embargo el protocolo propuesto, presenta un costo derivado de la comunicación entre los agentes que impacta en el tiempo necesario para la construcción del modelo.

Los resultados obtenidos demuestran que no hay diferencias significativas, en cuanto a la certeza en la clasificación, entre el modelo obtenido por el enfoque centralizado tradicional y el protocolo de aprendizaje colaborativo basado en agentes propuesto. La estrategia social logra una precisión equivalente a la obtenida con *J48* sin perder privacidad en los datos.

De acuerdo con las mediciones del tiempo necesario para construir el modelo, donde se comparan las dos estrategias sociales, la estrategia social mejorada resulta ser la solución factible, pues oculta mejor la información a las partes que participan en el proceso de aprendizaje colaborativo y, además toma menos tiempo para construir el modelo para algunos conjuntos de datos.

La estrategia propuesta ha demostrado no sólo ser factible, sin compartir los datos directamente, es capaz de construir modelos competitivos frente al enfoque centralizado tradicional al trabajar con datos verticalmente particionados. En Xavier Limón et al.[15] se propuso una solución de DDM para hacer frente a los datos particionados horizontalmente, aquí se amplió ese trabajo para que también pueda trabajar con datos heterogéneos.

El trabajo realizado en Chris Giannella et al[12] se centra más en la eficiencia en la comunicación cuando se construye el árbol, incluso cuando consiguen menos niveles de precisión que el enfoque centralizado tradicional y aquí, la estrategia colaborativa propuesta alcanza los mismos niveles de

precisión que la estrategia centralizada, no es una comparación justa, pues en este trabajo no se midió el costo de la comunicación debido a que el objetivo principal era lograr altos niveles de precisión y al mismo tiempo no compartir la información de las instancias entre los agentes. La eficiencia de la comunicación puede ser tomada en cuenta para el trabajo futuro, pero su mejora no se consideró en esta etapa de la investigación.

Un enfoque basado en agentes se hizo en Sung Baik y Jerzy Bala[3], para la inducción de árboles, pero sólo para dos agentes. En este trabajo se realizó algo similar pero se logró que funcione para n número de agentes. Fue probado con más agentes que atributos sin efectos adversos, los agentes sin atributos simplemente no toman parte en el proceso de aprendizaje.

5.2. Trabajo Futuro

El protocolo propuesto consigue niveles de certeza en la clasificación equivalentes a los obtenidos por la estrategia centralizada tradicional. En este trabajo se comunican los valores de *gain-ratio* e *info-gain* de todos los atributos debido al diseño del algoritmo de aprendizaje empleado. Esto nos lleva a algunas preguntas interesantes como, ¿cuántos atributos son necesarios para poder construir un modelo competitivo?, o ¿cuál es el resultado, si cada parte sólo envía los valores del mejor atributo local?. Para un trabajo futuro esto podría ponerse a prueba, así como el uso de algunos otros coeficientes estadísticos para determinar el mejor atributo, o la combinación de más de una métrica para elegir al ganador, así como tomar en cuenta otros paradigmas de aprendizaje.

En un trabajo futuro se puede diseñar otro protocolo social en el que el esquema es construido por los agentes participantes, será interesante probar diferentes estrategias de soporte de decisiones.

Las estrategias propuestas aquí suponen que la clase es bien conocida por cada parte. En Jaideep Vaidya et al. [26], las preocupaciones sobre la privacidad se toman más profundamente e incluso ésta no se comparte, ambos trabajos no utilizan el mismo algoritmo de aprendizaje, uno implementa *ID3* y el otro *J48*, pero las técnicas utilizadas para la preservación de la privacidad parecen interesantes para tener en cuenta para futuras mejoras de este trabajo.

Ocultar valores de los atributos y de la clase, así como implementar protocolos de seguridad para la comunicación de los datos entre los agentes, podrían ser probados con *J48*, este algoritmo implica más comunicación para construir el árbol que *ID3*, por lo que *J48* necesita compartir más.

Parte I

Apéndices

Apéndice A

Apéndice

Resumen: El siguiente artículo fue aceptado para presentación oral corta con poster y publicación en la treceava edición del MICAI (Mexican International Conference on Artificial Intelligence 2014), organizado por la SMIA (Sociedad Mexicana de Inteligencia Artificial), celebrado del 16 al 22 del noviembre de 2014 en Tuxtla Gutiérrez, Chiapas, teniendo como cede al Instituto Tecnológico de Tuxtla Gutiérrez y la Universidad Autónoma de Chiapas.

A.1. Publicación: MICAI-2014

Collaborative Data Mining on a BDI Multi-Agent System over Vertically Partitioned Data

Jorge Melgoza-Gutiérrez Alejandro Guerra-Hernández Nicandro Cruz-Ramírez
Universidad Veracruzana,
Departamento de Inteligencia Artificial,
Sebastián Camacho No 5, Xalapa, Ver., México 91000
jorge_melgoza@yahoo.com, {aguerra, ncruz}@uv.mx

***Abstract*—This paper presents a collaborative learning protocol, modeled and implemented under the Agents & Artifacts paradigm, to deal with heterogeneous training data, i.e., vertical partitions of examples. The artifacts encapsulate Weka [7] objects and methods that implement the learning algorithm to induce decision trees, based on a modified version of J48; while the agents manage the flow of the learning process. The proposed protocol is tested with well known training sets of the UCI [1] repository, comparing the obtained accuracy against the centralized, usual implementation of J48. Our collaborative learning protocol achieves equivalent accuracy to that obtained with J48, without losing privacy.**

I. Introduction

Traditional data mining algorithms were designed assuming the information is stored in a single source [6], that means all the learning process must be done in a centralized site. This view has changed with the widespread use of networks, where data is collected from different sources and is not necessarily stored in a single place; because of high storage requirements or limited bandwidth; and also because of privacy concerns do not allow sharing the information.

The traditional centralized approach is not an option to do a learning process under those conditions, in the other hand, Distributed Data Mining (DDM) can address this issues by performing data analysis and mining process, in a distributed manner that pays attention to these resource constraints [5]. Multi-Agent System (MAS) are, in essence, also distributed so these could work with DDM almost naturally.

DDM faces two kind of problems in a distributed relational database system, homogeneous data and heterogeneous data. Homogeneous data, also known as horizontal partitioning, means every part of the distributed system knows the scheme, but no one has all the samples. In the other hand, heterogeneous data or vertically partitioned data, means, in terms of instances and attributes, each site has different attributes (features) of the same samples (instances), one example would be the distinct areas in a hospital: weight, age, blood type, etc, could be gathered by one unit; but data from oncology tests in a different unit, but due the internal policies you are not allow to share information between units, maybe you want to do some learning process and get a model about cancer disease, but you do not want to tell who is sick because of privacy concerns.

Combining DDM and MAS could be a feasible solution for those kind of problems. This paper proposes a collaborative learning protocol based on the Agents & Artifacts paradigm, implemented with Jason [2] and CArtAgO [9], [10]. This work will focus on the problem of vertically partitioned data and at the same time, the privacy preserving concern will be taken into account, while performing data mining to build decision trees, but in no case the instances will be shared directly between agents. This work also provides an implementation for the experimental setting.

Another approach for DDM modeled and implemented under the Agents & Artifacts paradigm, also using the agent oriented programming language Jason [2] and CArtAgO [9], [10] JaCA-DDM is presented in [12], this is a solution for horizontally partitioned data, the work described in this paper will be an extension to handle vertically partitioned data.

An agent-based approach for tree induction over vertically partitioned data sets and its application to the computer network intrusion detection area, is proposed in [11], it uses the original ID3 algorithm developed by Quinlan [8] and is designed for two agents interacting through a mediator to build the decision tree. Their goal was to reduce the inter-agent communication bandwidth by finding ways to reduce the amount of information necessary for agent collaboration, in order to reduce the need of collecting the data from distributed hosts by applying a distributed data analysis on those distributed hosts.

Another proposal for vertically partitioned data [3], focused the work on the efficient construction of decision trees over heterogeneously distributed data, by doing a random projection-based dot product estimation and message sharing strategy. According to their experimental results this technique reduces the communication by a factor of five while still retaining 80% of the original accuracy.

More focused on privacy preserving, in [4] a generalized privacy-preserving variant of the ID3 [8] algorithm for vertically partitioned data distributed over two or more parties is presented. In this approach even the meta-data is concealed, the schema (attributes and their possible values) are protected from disclosure and the class is only know by one participant. The algorithm has been implemented on top of Weka [7] with modifications to allow one instance to be stored in different parties, also the internal process to classify an instance is modified to let it move from parties according to the tree, so even after the model is builded the attributes values are not

share.

For this work the concern will be to achieve accuracy levels highly competitive against the standard centralized approach, using the J48 algorithm, an open source Java implementation of the C4.5 decision tree algorithm, but the instances could not be share between members. In a future work more privacy preserving techniques could be tested. The collaborative learning protocol proposed here will work with vertically partitioned data where no one share the same attribute or attributes and the class is well known for all the parties, in this case agents.

This paper is organized as follows. Section 2 details the collaborative learning protocol. Section 3 explains particular details of the implementation. Section 4 presents the experimental settings. Section 5 shows the results obtained. Finally, Section 6 closes with the discussion and future work.

II. Learning Protocol

Decision tree building algorithms like J48, in order to build the tree, need to perform this steps (speaking generally): evaluate if there's a valid split and, if is the case, evaluate the attributes and choose the best. This valid split criterion could be for example enough instances to keep splitting the data. This lead us into two main choices: split the data and add another level to the tree; or make a leaf node. In a vertically partitioned data environment, these steps can be separated in two types: those you can do it just with your own data and those where you need all the data.

Agents can evaluate his own attributes but can't chose the best by their own, need to share some data to be able to compare all attributes. In the learning protocol proposed here, the agents will perform locally the attributes's evaluations and then, using a trusted third party (the *classifier artifact*), choose the best attribute to split.

A. Agents

Agents play two different roles in the learning process, the *learner* and the *worker*. All agents know the plans to perform each role, but to keep the order in the process, just one agent can be the *learner*.

1) Learner: The *agent learner* controls the learning process step by step, he decides when to ask for data to the workers, but this doesn't mean he can access all the data, his role is more like a guide. The *learner* evaluates the stop criterions of the algorithm and builds the leaf nodes when it's necessary. He also performs workers tasks when everybody does.

2) Worker: All agents do a worker's job. This kind of agents do the local evaluations with his local data, *information-gain* measures for example. When the best attribute is chosen the *worker* with that attribute is the one that builds the new node.

Following will be described the implemented artifacts to pass to the interaction through the learning protocol between the agents, and them with the artifacts, in order to make it more clear to understand.

B. Artifacts

An artifact is a first order abstraction used for modeling computational environments. Each artifact represents an entity of the environment, offering services (operations) and data structures (observable properties) to the agents, in order for them to improve their activities. Artifacts are conceived as function-oriented computational devices, agents may exploit this functionalities in order to fulfill their individual and social activities.

For the learning process proposed here, there are two main artifacts: *classifier* and *evaluator*, each agent has an *evaluator artifact* but there is only one *classifier artifact*.

1) Classifier Artifact: This artifact encapsulates the tree structure and provides operations for agent coordination and attribute selection, gets the attributes's evaluations and new nodes to attach to the tree.

1) Observable property, *nodeStatus*, reveals the current node stage.
   a) 0 Means it's a new node so the agents will perform a learning cycle.
   b) 1 The current node has a child to be processed.
   c) 2 Leaf node or each child has been already processed.
   d) 3 When the backtrack reaches the root so the tree is complete.
2) Operations
   a) backtrack. Moves to current node's father, update *nodeStatus* and send a *backtrack* signal to all agents.
   b) checkValidM. Check the enough-valid-models flag.
   c) findBest. Chose the best attribute and check if useful split was found.
   d) getValues. For social strategy, allows the agent to put the attributes's evaluations inside the object.
   e) processAvInfG. Calculate average *info-gain*.
   f) sendRequest. Send *requestatt* signal to the winner agent.
   g) setReady. For coordination between agents. Sends a signal when all agents notify that they are ready.

2) Evaluator Artifact: This artifact encapsulates Weka [7] objects and methods to evaluate the attributes and build the nodes. It controls the stack of data partitions (add and drop). Also sends the evaluations and nodes to the *classifier artifact*. This artifact has no observable properties due the privacy preserving concerns.

1) Operations
   a) checkMultival. Check if at least one attribute is nominal and have a lot of values.
   b) drop. Remove last partition from the stack.
   c) evalAttributes. Evaluate all local attributes from the last partition in the stack.
   d) evalSplit. Check if all Instances belong to the same class or if not enough Instances to do the split.

e) getTrainData. Create the stack and set the first partition.
f) sendAttibute/sendNosplit. Send a new node according to the attribute selection results. If there is a valid split it builds the node with the winner attribute, if not Nosplit builds a leaf node.
g) sendMultival. Update multival-flag according to the results of all participants.
h) sendValues. Send attributes's evaluations to the *classifier artifact*.
i) setTraintmp. Creates new partitions and adds these to the end of the stack.

C. Learning Process

Figure1 shows the interaction between agents and artifacts through the learning process.

Here is a more detailed explanation of this steps:

1) The agent *learner*, using the *evaluator artifact* verifies the stopping criterion. All instances of the current partition belong to the same class, or there is not enough instances to perform a valid partition (2 minimum according to J48), either one of this is true it means is a leaf node.
2) Next a flag is validate for all agents according to J48, it becomes false if at least one attribute is nominal and has many values. The result is shared by the agents through the *classifier artifact*.
3) Then the agent *learner*, asks everyone to evaluate their attributes using the methods encapsulated in their own *evaluator artifact*, the results are send to the *classifier artifact*. Each agent sends a list made by the agent's name, an local identifier for his attributes (not the attribute name, in this case an integer) and the values, then tells he is ready trough the *classifier artifact*. [1]
4) When every agent notifies he is ready, a signal is emitted and the agent *learner* proceed to calculate the *total valid models* and *average information gain* values. If there is no valid models, the stopping criterion of the algorithm is reach and thus a leaf node was found.
5) After the calculations the agent *learner* uses the *classifier artifact* to establish which is the best attribute, then a signal is sent to the corresponding agent telling him the attribute's identifier.
6) The agent with the best attribute uses his *evaluator artifact* to create the corresponding node and send it to the *classifier artifact*, finishing this cycle with a signal to everyone.

Each time a leaf node is builded a signal is emitted so all agents update their partitions for the next cycle.

[1]Two solutions were proposed to test the protocol: one called *social* approach, where the *workers* pass directly to the *learner* the results of the attributes's evaluations by the use of speech acts, and the *learner* put those in the *classifier artifact* to do the attribute selection; in the other hand in the called *improved social* approach, no direct communication of the attributes's evaluations takes place, instead, the data is sent through artifact connections so no one share directly the information.

III. Implementation Details

The agent *learner* controls the learning process workflow and at certain stages requests information to other agents, such as for example, the *information gain*, so is not possible to run the procedure recursively because each agent does the calculations locally.

As is known J48, been an extension of the original ID3 algorithm, build the tree by Depth-first search. The recursion allows each new call to get the corresponding data partition that will be processed. In order to do the same, the collaborative learning strategy proposed implements a stack, where the partitions will be managed.

Each agent initializes his stack with his entire local database. Each cycle, all agents will take the last element from his stack to perform the attribute evaluation. If the agent *learner* that coordinates the process, determines that it has been found a valid partition, all agents will be notified with a list of the indexes corresponding to the instances belonging to the new partition or partitions made (it can be a list of lists), according to the selected attribute.

Next each agent will send to the end of his stacks the new partitions made using the last partition of the stack, according to the lists received. If the agent *learner* determines that there is no valid partition, it means is a leaf node, so the agents will be notified to drop the last element from the stack like doing a *backtrack*. The *backtrack* process is also call after all children from a node are processed.

Weka [7] was made for centralized processing, so another modification was made. Internally when the tree is builded, to identify the corresponding attribute for each node, only an integer is stored in the structure, because is expected there will be only one scheme and the attributes will not change position. But when this is moved to the distributed environment, every party could have an attribute identified by the same number 1, 2, etc.

To avoid this miss reference to attributes, the internal process was modified so also the name of the attribute is internally stored by the tree, no agent can see it because is only for inside processing when classifying an instance, but that way, when the classify process is called each node refers to the proper attribute.

IV. Experimental Design

Table I contains the databases used for the experiments reported, all get from the UCI [1] repository.

Stratified cross validation with 10 folds and 10 repetitions was applied. For each database, experiments were done with 2, 4, 7, 10, 15 and 20 agents, do the number of attributes only when there is enough for at least one for each agent.

JaCA-DDM [12] provides a tool to setup the experiments, it creates random stratified data partitions and distributes them among the agents, but is designed for horizontally partitioned data. This work of loading the data set and make all the partitions for the experiment is done by the *oracle artifact*. It loads an ARFF file and creates random partitions according to the number of agents that will take place in the learning process.

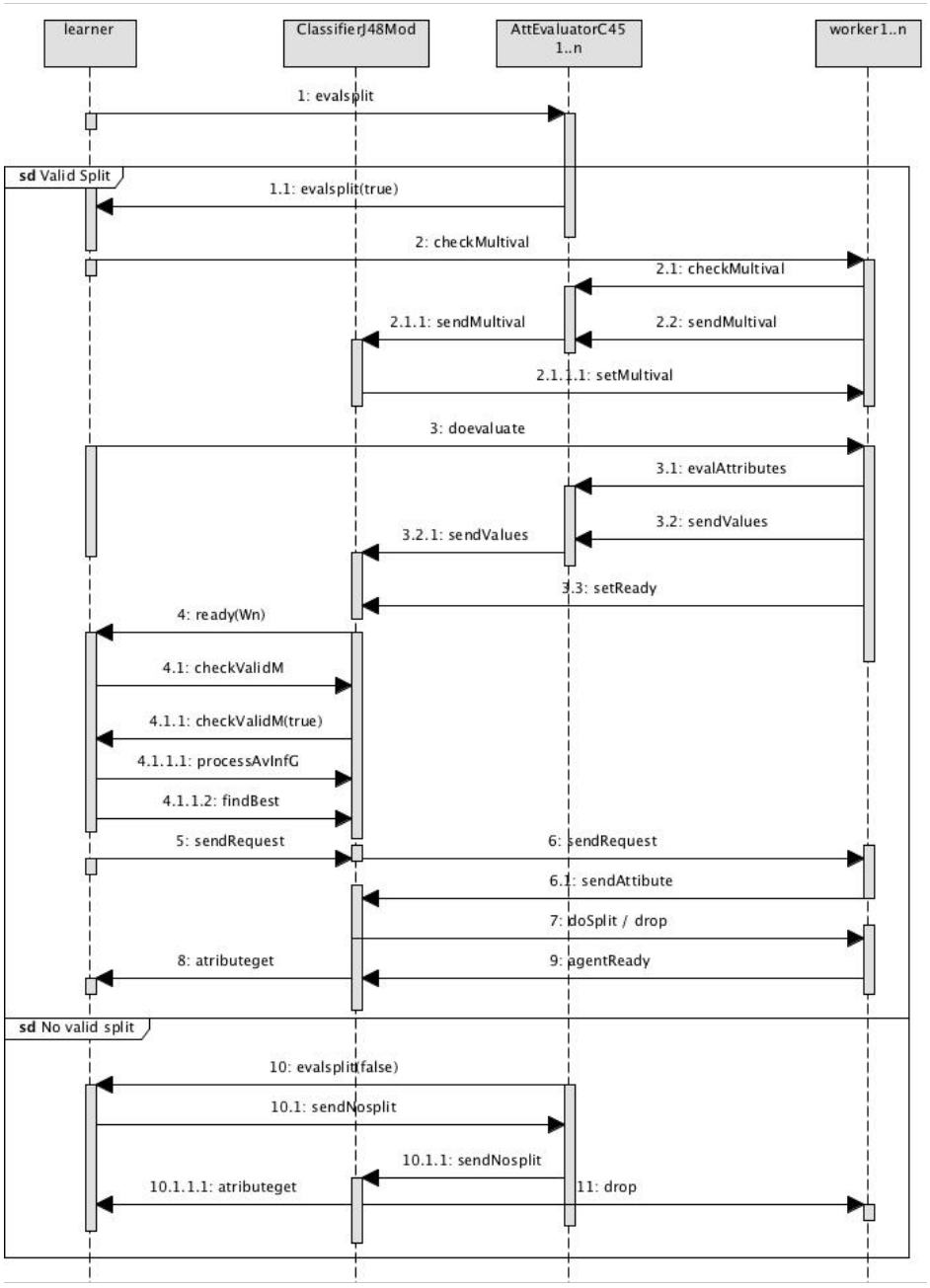

Fig. 1. Learning process interaction

TABLE I. DATA SETS

Data Set	Instances	Attributes	Classes
australian	690	14	2
breast	683	9	2
car	1728	6	4
german	1000	20	2
heart	270	13	2
iris	150	4	3
letter	20000	17	26
mushroom	8124	22	2
waveform	5000	40	3

TABLE II. ACCURACY RESULTS

Data Set	#A	Centralized	Social	Improved Social
australian	2	83.6377	83.6667	83.7101
australian	4	83.6377	83.5797	83.6377
australian	7	83.6232	83.5652	83.7681
australian	10	83.7246	83.6812	83.6667
breast	2	95.6215	95.7089	95.6503
breast	4	96.0934	96.0786	96.0782
breast	7	96.0735	96.0586	96.1172
car	2	93.4785	93.3685	93.3743
car	4	93.7211	93.6343	93.6575
german	2	68.2600	68.1400	68.1900
german	4	68.8100	68.6100	68.6700
german	7	68.4300	68.4700	68.5500
german	10	69.1600	69.1300	69.1500
german	15	68.6300	68.6100	68.6200
german	20	68.8000	68.9000	68.7000
heart	2	77.1111	77.1111	77.0741
heart	4	77.4444	77.5185	77.6667
heart	7	77.2963	77.5926	77.4074
heart	10	77.3333	76.9259	77.3333
iris	2	93.8000	93.7333	93.6000
iris	4	94.8667	94.8000	94.8000
letter	2	88.0065	88.0295	88.0210
letter	4	88.0775	88.0835	88.0575
letter	7	88.1155	88.1200	88.1025
letter	10	87.9380	87.9295	87.9520
letter	15	87.9825	87.9825	88.0105
mushroom	2	100.0000	100.0000	100.0000
mushroom	4	100.0000	100.0000	100.0000
mushroom	7	100.0000	100.0000	100.0000
mushroom	10	100.0000	100.0000	100.0000
mushroom	15	100.0000	100.0000	100.0000
mushroom	20	100.0000	100.0000	100.0000
waveform	2	75.2400	75.3220	75.3720
waveform	4	75.2040	75.1500	75.3240
waveform	7	75.5400	75.6520	75.5960
waveform	10	75.3740	75.4080	75.4460
waveform	15	75.3460	75.5020	75.5160
waveform	20	75.5900	75.4300	75.4940

This artifact was modified so it could also create vertically partitioned data, to do this it makes a shuffled list of attributes (list of integers) and using a roulette gives one by one the attributes to the *n* partitions for the *n* agents, this list is saved so all the strategies, the centralized and social ones, get the same scheme each cycle, for the cross validation method each fold also gets the same partitions, just the instances for the train and test partitions do change.

V. RESULTS

The Table II reports accuracy measures obtained for each different model. It shows the mean for every data-set and number of agents combination. The tailed paired T test with 0.05 degrees of significance was also calculated to verify if there are significant differences between the strategies. 0 means there is not significant difference; -1 means that the first strategy paired won; and 1 means that the first strategy lost. Obtained data from the tailed paired T test is not show in the table because all comparisons gave 0 as result.

TABLE III. TIME MEASURING RESULTS

Data Set	#A	Centralized	Social	Improved Social	SvsIS
australian	2	8.15	6,848.25	5,248.68	0
australian	4	7.75	6,880.79	5,483.51	0
australian	7	7.85	8,881.66	9,328.11	0
australian	10	7.89	7,339.87	7,490.95	0
breast	2	3.85	2,251.65	2,186.84	0
breast	4	4.08	3,018.82	2,595.51	0
breast	7	4.55	4,704.22	4,558.54	0
car	2	6.26	6,608.78	5,434.17	0
car	4	6.43	8,611.05	7,747.26	0
german	2	12.75	12,843.72	7,140.47	1
german	4	13.02	11,278.33	9,318.01	0
german	7	13.98	32,178.03	32,221.80	0
german	10	14.15	24,781.82	25,154.03	0
german	15	15.27	59,475.05	57,788.06	0
german	20	15.61	124,261.13	128,091.83	0
heart	2	3.99	3,080.62	1,702.95	1
heart	4	4.41	4,148.98	3,676.86	0
heart	7	4.59	5,269.74	5,508.55	0
heart	10	4.66	4,188.10	3,797.46	0
iris	2	3.48	775.19	698.34	0
iris	4	3.79	711.56	771.53	0
letter	2	3,386.61	97,067.45	80,622.19	1
letter	4	3,454.92	151,577.55	122,904.79	1
letter	7	3,441.29	196,337.16	178,184.22	1
letter	10	3,373.14	181,797.80	170,953.10	1
letter	15	3,454.74	398,810.68	352,934.27	1
mushroom	2	30.07	1,302.95	1,124.56	0
mushroom	4	31.82	1,021.48	957.93	0
mushroom	7	32.85	1,838.37	1,877.59	0
mushroom	10	34.29	2,229.76	2,245.37	0
mushroom	15	36.21	3,703.65	2,040.21	1
mushroom	20	36.86	7,814.06	5,365.93	1
waveform	2	808.59	20,076.27	15,279.44	1
waveform	4	817.30	23,973.18	18,600.12	1
waveform	7	821.73	33,587.63	31,784.22	0
waveform	10	825.37	26,397.31	21,177.41	1
waveform	15	828.80	33,652.16	22,976.08	1
waveform	20	833.82	105,877.91	95,614.76	1

The time needed to build the model was also measured. In this stage the main focus was to achieve competitive accuracy levels, the time is relevant just to compare the performance between the two social strategies but not against the centralized approach, because there is no transmission cost do the concurrent processing, this times are show in Table III, the scale is in milliseconds and the last column represents the tailed paired T test results for the social vs improved social comparison. The results comparing the centralized approach against the two social strategies gave -1.

The centralizing process takes less time to build the model, but this is expected because each cycle of the learning algorithm, the centralized approach only consumes time to evaluate the attributes and choose the best among them, but the social approach also needs time for a reasoning cycle of each agent, in the better case only the longer reasoning cycle from one of the agents (completely parallel or concurrent) but the worst case could be a sum of them (not concurrent). The results show mostly no significant differences, in the time measured, between the two social strategies, and when there is, it favors the improved social approach.

According to the time measures comparing the two social strategies, the improved social strategy proves to be a feasible solution, it conceals better the information to the learning parties than the social strategy and also take less time to build the model for some data sets.

VI. Discussion and Future Work

The results shown in Table II prove there is no significant differences on accuracy between the model obtained by the traditional centralized approach and the strategy proposed in this paper, so the collaborative learning protocol achieves equivalent accuracy to that obtained with J48, without losing privacy.

The agents & artifacts paradigm, proved to be a very helpful tool, the particular methods for attribute evaluation are enclosed inside the artifact, so the learning protocol is not linked to an specific metric or method to do this evaluation.

The strategy proposed has proven not only feasible but also, without sharing the data directly, able to build models highly competitive against the centralized traditional approach while working with vertically partitioned data. In [12] an approach for DDM was proposed to deal with horizontally partitioned data, here this was enhanced so it can also work for vertically partitioned data, another improve so it can work over data partitioned both horizontally and vertically will be interesting in a future work.

Even when the measured times not involve real transmission cost, the improved social approach is slightly faster than the social approach, so it shows that for data communication, the link between artifacts is more efficient than the communication between agents and its better in terms of security do the fact it only allows the agents to see certain predefined information not all the data in the artifact.

An agent-based approach was made in [11] for tree induction but only for two agents, in this paper a similar work was done but achieving to make it work with n number of agents, it was tested with more agents than attributes with no side effects, the agents with no attributes just do not take part in the learning process.

The work done in [3] is more focused on efficiency on communication when constructing the tree, even when they get less accuracy levels than the traditional centralized approach and here the collaborative strategy proposed achieve same accuracy levels, direct comparison its not fair, in this work the cost of communication was not measured because the main focus was to achieve the accuracy levels and at the same time do not share the information between agents, the efficiency of communication can be taken on account for future work but it was not considered at this stage of the investigation.

The strategies proposed here assume the class is well known for each party, in [4] the privacy concerns are taken more deeply and even that is not shared, both works use not the same learning algorithm, one implements ID3 and the other J48, but the techniques used for privacy preserving appear interesting to take into account for future improvements of this work. Conceal the attributes and class values, as well as security protocols for the data communication between agents could be tested with J48, this algorithm implies more communication to build the tree than ID3, so J48 needs to share more.

This lead us to some interesting questions like, how many attributes are needed to be able to build a competitive model?, in this case the gain-ratio and info-gain from all attributes is communicated, what is the result if each party just sends the best local attribute?. For future work this could be tested, also the use of some other statistical ratios to determine the best attribute, and combining more than one metric.

In this work all the experiments were done in a concurrent environment, so even when the time for each learning process was measured, it is not relevant data because the transmission costs were near to zero, it will be interesting to do more test in a real distributed environment, to see the influence of traffic and other network related problems on the learning process times and performance.

Also design another social protocol where the scheme is builded by the participant agents could be done in a future work, will be interesting to test different decision support strategies.

References

[1] Bache, K., Lichman, M.: UCI machine learning repository (2014)

[2] Bordini, R.H., Hbner, J.F., Wooldridge, M.: Programming multi-agent systems in Agent Speak using Jason, vol. 8. Wiley-Interscience (2007)

[3] C. Giannella, K. Liu, T. Olsen, and H. Kargupta. Communication Efficient Construction of Decision Trees Over Heterogeneously Distributed Data. In Proceedings of the Fourth IEEE International Conference on Data Mining, pages 6774, (2004)

[4] Jaideep Vaidya, Chris Clifton, Murat Kantarcioglu, and A. Scott Patterson. 2008. Privacy-preserving decision trees over vertically partitioned data. ACM Trans. Knowl. Discov. Data 2, 3, Article 14, 27 pages.(2008)

[5] Josenildo C. da Silva, Chris Giannella, Ruchita Bhargava, Hillol Kargupta, and Matthias Klusch. 2005. Distributed data mining and agents. Eng. Appl. Artif. Intell. 18, 7 , 791-807, (2005)

[6] Li Zeng et al, "Distributed data mining: a survey", Springer US, Volume 13, Issue 4, pp 403-409.(2012)

[7] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten; The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1. (2009)

[8] Quinlan, J. R., and Rivest, R.L., Inferring Decision Trees Using the Minimum Description Length Principle, Information and Computation, Vol. 80, no. 3, (1989)

[9] Ricci, A., Piunti, M., Viroli, M.: Environment programming in multi-agent systems: an artifact-based perspective. Autonomous Agents and Multi-Agent Systems 23(2), 158192 (2011)

[10] Ricci, A., Viroli, M., Omicini, A.: Construenda est CArtAgO: Toward an infrastructure for artifacts in MAS. Cybernetics and Systems 2, 569574 (2006)

[11] S. Baik, J. W. Bala, A decision tree algorithm for distributed data mining: Towards network intrusion detection, in: ICCSA, (2004)

[12] Xavier Limn; Alejandro Guerra Hernndez; Nicandro Cruz Ramrez; Francisco Grimaldo, F. Castro, A. Gelbukh, M.G. Mendoza (Eds.): MICAI 2013, Part II, LNAI 8266, pp. 338349, 2013. Springer-Verlag Berlin Heidelberg (2013)

Referencias

Locura es hacer lo mismo una vez tras otra y esperar resultados diferentes.

Albert Einstein

[1] Aristóteles. *Prior Analytics No 391. En: Loeb Classical Library.* Harvard University Press, 1960.

[2] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[3] Sung Baik and Jerzy Bala. A decision tree algorithm for distributed data mining: Towards network intrusion detection. In Antonio Laganá, MarinaL. Gavrilova, Vipin Kumar, Youngsong Mun, C.J.Kenneth Tan, and Osvaldo Gervasi, editors, *Computational Science and Its Applications - ICCSA 2004*, volume 3046 of *Lecture Notes in Computer Science*, pages 206–212. Springer Berlin Heidelberg, 2004.

[4] Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge. *Programming Multi-Agent Systems in AgentSpeak Using Jason*, volume 8. Wiley-Interscience, 2007.

[5] Michael E. Bratman. *Intention, Plans, and Practical Reason.* Cambridge, MA., USA, y London, England: Harvard University Press, 1987.

[6] Rodney A. Brooks. *Cambrian Intelligence: The Early History of the New AI.* MIT Press, Cambridge, MA, USA, 1999.

[7] Arie A. Covrigaru and Robert K. Lindsay. Deterministic autonomous systems. *AI Mag.*, 12(3):110–117, September 1991.

[8] Josenildo C. da Silva, Chris Giannella, Ruchita Bhargava, Hillol Kargupta, and Matthias Klusch. Distributed data mining and agents. *Eng. Appl. Artif. Intell.*, 18(7):791–807, October 2005.

[9] Daniel C. Dennett. *The Intentional Stance.* MIT Press, 1987.

[10] Daniel Clement Dennett. Intentional systems. *The Journal of Philosophy 68.4*, pages 87–106, 1971.

[11] Oren Etzioni. Intelligence without robots: A reply to brooks. *AI Mag.*, 14(4):7–13, December 1993.

[12] Chris Giannella, Kun Liu, Todd Olsen, and Hillol Kargupta. Communication efficient construction of decision trees over heterogeneously distributed data. In *Proceedings of the Fourth IEEE International Conference on Data Mining*, ICDM '04, pages 67–74, Washington, DC, USA, 2004. IEEE Computer Society.

[13] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

[14] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[15] Xavier Limón, Alejandro Guerra-Hernández, Nicandro Cruz-Ramírez, and Francisco Grimaldo. An agents and artifacts approach to distributed data mining. In Félix Castro, Alexander Gelbukh, and Miguel González, editors, *Advances in Soft Computing and Its Applications*, volume 8266 of *Lecture Notes in Computer Science*, pages 338–349. Springer Berlin Heidelberg, 2013.

[16] Wolfram Müller-Freienfels. *"Agency". En: Encyclopedia Britannica. Internet version*. Encyclopedia Britannica, Inc., 1999.

[17] Claude Nadeau and Yoshua Bengio. Inference for the generalization error. *Machine Learning*, 2001.

[18] Andrea Omicini, Alessandro Ricci, and Mirko Viroli. Artifacts in the a&a meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3):432–456, December 2008.

[19] J. R. Quinlan and R. L. Rivest. Inferring decision trees using the minimum description length principle. *Inf. Comput.*, 80(3):227–248, March 1989.

[20] Anand S. Rao and Michael P. Georgeff. Bdi agents: From theory to practice. In *IN PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON MULTI-AGENT SYSTEMS (ICMAS-95*, pages 312–319, 1995.

[21] Vuda Sreenivasa Rao. Multi agent-based distributed data mining: An overview. *International Journal of Reviews in Computing*, 3:83–92, 2009.

[22] Alessandro Ricci, Michele Piunti, and Mirko Viroli. Environment programming in multi-agent systems: An artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, 23(2):158–192, September 2011.

[23] Alessandro Ricci, Mirko Viroli, and Andrea Omicini. Construenda est cartago: Toward an infrastructure for artifacts in mas. In *CYBERNETICS AND SYSTEMS 2006*, pages 569–574, 2006.

[24] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[25] John R. Searle. What is an intentional state? *Mind*, 88(January):74–92, 1979.

[26] Jaideep Vaidya, Chris Clifton, Murat Kantarcioglu, and A. Scott Patterson. Privacy-preserving decision trees over vertically partitioned data. *ACM Trans. Knowl. Discov. Data*, 2(3):14:1–14:27, October 2008.

[27] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10:115–152, 1995.

[28] Li Zeng, Ling Li, Lian Duan, Kevin Lu, Zhongzhi Shi, Maoguang Wang, Wenjuan Wu, and Ping Luo. Distributed data mining: a survey. *Information Technology and Management*, 13(4):403–409, 2012.