
Exploraciones sobre el soporte Multi-Agente en Minería de Datos

Alejandro Guerra-Hernández, Nicandro Cruz-Ramírez, and Rosibelda Mondragón-Becerra

Departamento de Inteligencia Artificial
Facultad de Física e Inteligencia Artificial
Universidad Veracruzana
Sebastián Camacho No. 5, Xalapa, Ver.,
México 91000
{aguerra, ncruz, rmondragon}@uv.mx

1. Introducción

Desde el fundacional *The Art of Human Computer Interface Design* [16], el concepto de agente interfaz inteligente ha estado presente en la agenda de la Inteligencia Artificial (IA) y no exento de debate (manipulación directa *versus* delegación). A lo largo de estos años, la investigación en Agentes y Sistemas Multiagentes (SMA) se ha establecido como un área de investigación interdisciplinaria [13, 26, 28], con un fuerte anclaje en la Inteligencia Artificial Distribuida. Dentro de nuestro grupo de investigación, hemos ido de la concepción de agentes interfaz inteligentes bajo la metáfora de asistente personal y diseñados bajo el precepto de delegación [1], al estudio de los SMA para el soporte de procesos interactivos, como el Descubrimiento de Conocimiento en Bases de Datos (KDD), que consideran no sólo delegación, sino comunicación, colaboración y demás mecanismos propios de los agentes racionales.

Una de las principales metas del KDD, es proveer métodos capaces de encontrar patrones, regularidades y conocimiento implícito en los datos, de forma que sea posible entender el fenómeno estudiado [11]. Puesto que la cantidad de datos crece muy rápido, resulta urgente proponer nuevos enfoques que permitan procesar y manipular estos datos de una manera rápida y confiable. Para ello, el KDD combina ideas y técnicas de diversas áreas como las bases de datos, la estadística, el aprendizaje automático y la IA, entre otras.

El proceso de KDD consta de diferentes etapas que van de la selección, preprocesamiento y transformación de los datos, a la extracción de patrones, el paso conocido como Minería de Datos, así como su evaluación e interpretación. Cada paso involucra diversas técnicas, siendo muchos de ellos áreas de investigación por si mismos. Nuestro interés esta focalizado en el paso de Minería de Datos, más específicamente en la tarea conocida como clasificación.

El problema de clasificación puede definirse brevemente como sigue: Dados un conjunto de casos sin etiquetar y un conjunto de etiquetas, el problema consiste en encontrar una función que mapee de manera adecuada cada caso no etiquetado con su etiqueta correspondiente o clase. En otras palabras, una función que clasifique los casos sin etiquetar correctamente, usando como clases el conjunto de etiquetas dado. El diseño de clasificadores automáticos que estimen esta función a partir de los casos sin etiquetar (datos), es de interés capital. Como resultado, tenemos diferentes métodos de clasificación como los basados en la regresión, los árboles de decisión, las redes Bayesianas y las redes neuronales, entre otros [11, 27]. Así como la implementación de estos algoritmos en paquetes de software como Clementine, DBMiner, Weka, etc. [11, 27, 24].

Weka se ha convertido en uno de los sistemas KDD más populares, debido a que provee muchos de los procedimientos involucrados en el proceso completo de KDD. Es importante señalar que aunque Weka permite ahorrar mucho tiempo en el preprocesamiento, análisis y evaluación de los datos, uno debe correr todos sus algoritmos manualmente. Por ejemplo, si queremos probar el desempeño de diferentes clasificadores en varias bases de datos, el preprocesamiento puede incluir la discretización de las variables continuas en todas las bases de datos. Peor aún, el usuario dispone de alrededor de sesenta métodos de clasificación a elegir. Finalmente, existen muchas combinaciones de los algoritmos en estos tres módulos (preprocesamiento, clasificación y evaluación), más el número de bases de datos a ser analizados. Por lo tanto un meta-clasificador automático [25] que nos permita ver a los sistemas de aprendizaje como agentes racionales capaces de adaptarse a ambientes específicos, explotando el conocimiento ganado por sus experiencias, sería más que deseable.

Aquí es donde entra nuestra propuesta: diseñar e implementar un SMA de agentes racionales BDI [10, 21, 28], como soporte para decidir que combinación de técnicas es la mejor. Los resultados preliminares en esta dirección, nos muestran la posibilidad y conveniencia de un SMA que integre partes de un proceso KDD utilizando dos clasificadores clásicos: Naives Bayes (NB) y C4.5 [8, 9, 20]. Los resultados indican que es posible construir de manera natural un meta-clasificador basado en este marco SMA. Si bien el uso de los SMA en procesos KDD no es del todo novedoso [14, 29, 15, 17], nuestra preferencia por los agentes racionales BDI [28] y su metodología asociada si lo son. Este paradigma nos provee con el nivel adecuado de abstracción y las herramientas de implementación necesarias para este proyecto.

El resto del artículo está organizado como sigue: La sección 2, introduce el concepto de agente BDI con base en AgentSpeak(L) [21] y al intérprete Jason [3], la implementación de AgentSpeak(L) que hemos elegido para estas pruebas. La sección 3 reseña brevemente el proceso de KDD usando Weka [27]. La sección 4 discute el diseño e implementación de un prototipo de SMA operando como soporte en Weka para el uso de dos clasificadores: Naives Bayes [9] e ID3 [20]. Finalmente, la sección ofrece las conclusiones de estos primeros experimentos.

2. Agentes Intencionales: Jason y AgentSpeak(L)

Entre los modelos de agencia racional propuestos en IA, el modelo de agentes intencionales BDI (*Belief-Desire-Intention*) ha resultado de gran relevancia. Esto obedece a que el modelo cuenta con sólidos fundamentos filosóficos, basados en la actitud intencional de D. Dennett [6] y la teoría del planes, intenciones y razonamiento práctico de M. Bratman [4]. Estas dos nociones de intencionalidad nos proveen con las herramientas para describir los agentes a un nivel adecuado de abstracción, al adoptar la actitud intencional, y definirlos funcionalmente de manera compatible con tal actitud, como sistemas basados en razonamiento práctico. Ambos aspectos de la intencionalidad han sido formalmente expresados y estudiados bajo diferentes lógicas de elegante semántica [22, 23, 28], entre las que destaca AgentSpeak(L) [21]. Existen además diferentes implementaciones del modelo BDI, como IRMA [5], y los sistemas *à la* PRS: PRS [10], dMARS [7] y Jam [12]. Bordini et al. [2], ofrecen una revisión más extensa de los lenguajes y plataformas disponibles para programar agentes BDI.

Para la realización de este proyecto, hemos optado por el modelo formal propuesto en AgentSpeak(L), debido a su sencillez, su claridad y su semántica operacional bien definida [18]. Para lograr compatibilidad con Weka, hemos optado por un intérprete escrito en Java, conocido como Jason [3]. A continuación introducimos ambas elecciones.

2.1. AgentSpeak(L)

Cuando los enfoques para implementar agentes intencionales se orientaban principalmente hacia el diseño de arquitecturas computacionales, A. S. Rao abrió una alternativa más cercana a la programación orientada a agentes con AgentSpeak(L) [21]. Su idea central era proveer una caracterización concisa en lógica de primer-orden de los agentes BDI, inspirada en la programación lógica. Su propuesta incluía un lenguaje para programar agentes, con base en los siguientes constructores:

- Creencias: Literales de base de la lógica de primer-orden. Constituyen la representación que cada agente tiene de su entorno.
- Metas: Se reducen a preguntarse (?) si una literal es verdadera (*test goal*) dadas las creencias del agente; y a lograr (!) que una literal se vuelva verdadera (*achieve goal*) a través de la actuación de los agentes.
- Eventos disparadores: El propósito de un agente es observar su medio ambiente, y con base en sus observaciones y sus metas, ejecutar ciertas acciones. Los agentes en AgentSpeak(L) son especialmente sensibles a los llamados eventos disparadores (*trigger events*), que cubren alteraciones en las creencias y metas del agente. Agregar (+) o eliminar (-) creencias y metas, constituyen las observaciones que mueven a un agente a elegir sus cursos de acción.
- Acciones: Las acciones primitivas son procedimientos que al ejecutarse cambian el estado del medio ambiente donde se encuentra el agente. Se representan por un nombre y un conjunto de parámetros que toman la forma de términos de la lógica de primer orden.

- Planes: Toman la forma siguiente: *evento disparador : contexto <- cuerpo*. El contexto es una conjunción de literales, que debe observarse como verdadera para que el plan pueda ser adoptado. El cuerpo del plan es una secuencia de metas o acciones primitivas.

A continuación se muestra un ejemplo de plan:

```
+!aprenderModelo(X): true
  <- .print("Aprendiendo modelo ID3...");
  weka.evaluarModelo(X,N);
  !verifica(X,N);
  !filtraRes(X,N).
```

La lectura de este plan es la siguiente. Si se percibe un evento disparador que indica que la meta *!aprenderModelo(X)* ha sido agregada al estado del agente, para cierto valor de la variable *X*; entonces, en cualquier caso (contexto *true*), ejecutar el cuerpo del plan secuencialmente: imprimir “Aprendiendo modelo ID3”, ejecutar la acción primitiva *weka.evaluarModelo(X,N)* y registrar dos nuevas (sub) metas: *!verifica(X,N)* y *!filtraRes(X,N)*. La notación para las acciones primitivas asume un formato *librería.acción*. En este ejemplo, utilizamos la acción primitiva *evaluarModelo* de la librería *weka*. El contexto del plan puede ser cualquier conjunción de literales, no necesariamente de base.

Un agente BDI consiste en una tupla $\langle E, B, P, I, A, S \rangle$ (configuración) de conjuntos de eventos, creencias, planes, intenciones, acciones y funciones de selección, respectivamente. Donde las intenciones, son pilas de planes parcialmente definidos, cuya adopción responde a la percepción y metas del agente. AgentSpeak(L) define una teoría de prueba basada en un sistema de transición $\langle \Gamma, \vdash \rangle$ donde Γ es un conjunto de configuraciones BDI y \vdash una relación binaria de entre ellas. Un algoritmo simplificado de la operación de un agente BDI, sería como sigue (las funciones *selEvento*, *selAplicable*, *selRelevante*, *selIntencion* $\in S$):

```
1: while true do
2:    $E \leftarrow \text{percepcion}()$ 
3:   while  $E \neq \emptyset$  do
4:      $e \leftarrow \text{selEvento}(E)$ 
5:      $p \leftarrow \text{selAplicable}(\text{selRelevante}(B, P, e))$ 
6:      $I \leftarrow \text{actualiza}(I, p)$ 
7:   end while
8:    $\text{ejecuta}(\text{selIntencion}(I))$ 
9: end while
```

El trabajo original de A. S. Rao provee reglas de transición para intentar un meta, intentar una submeta y ejecutar una acción. Á. Moreira y R. Bordini [18] proveen la semántica operacional completa para AgentSpeak(L), como base para el intérprete Jason [3].

2.2. Jason

Jason [3] implementa en Java, una versión extendida de AgentSpeak [18]. El intérprete incluye actos de habla, basados en KQML, para la comunicación entre los agentes; anotaciones en los planes para utilizar funciones de selección basadas en teoría de la decisión; la posibilidad de distribuir el SMA en varias computadoras sobre una red; funciones de selección configurables en Java; mecanismos de extensión con base en la definición de nuevas acciones primitivas; y facilidades para la programación del medio ambiente del SMA. Si no se consideran estas extensiones, Jason es casi idéntico al elegante AgentSpeak(L).

El ambiente integrado de desarrollo de Jason, provee una interfaz gráfica para editar las configuraciones SMA y el código de los agentes escrito en AgentSpeak(L). Usando este ambiente, es posible controlar la ejecución del SMA y distribuir a los agentes sobre una red de manera sencilla. De especial utilidad, resulta el inspector del ambiente, que permite observar el estado interno de los agentes en tiempo de ejecución.

El modelado de los SMA implementados con Jason, pueden realizarse utilizando la metodología Prometheus y su herramienta asociada PDT [19]. PDT ofrece soporte para las tres fases de diseño especificadas en Prometheus:

- Especificación de sistema: Se identifican las metas del sistema; se captura la relación entre los agentes y el medio ambiente en términos de acciones y percepciones; se describen las funcionalidades deseadas; y se desarrollan los escenarios que los agentes pueden confrontar.
- Arquitectura: Se especifican los tipos de agentes en el sistema combinando las funcionalidades obtenidas en el paso anterior; la estructura general del sistema se describe usando una gráfica (como la mostrada en la figura 2); y se capturan los protocolos de interacción en términos de secuencias legales de mensajes.
- Detalles: Se especifican los aspectos internos de cada agente, en términos de capacidades, eventos, planes y datos.

3. Descubrimiento de Conocimientos con Weka

En la introducción mencionamos brevemente los pasos que involucra el proceso KDD. En esta sección, veremos con un poco más de detalle dichos pasos usando Weka. La figura 1 muestra las etapas del proceso y la naturaleza iterativa de éste. Como se puede observar, la meta última es la de conformar nuevo conocimiento a partir de los datos provenientes de algún dominio específico. Sin embargo, lograr esta meta no es una tarea fácil pues involucra varias tareas que tienen, en general, un alto grado de complejidad. Por mencionar un ejemplo, la recolección y selección de los datos de un fenómeno perteneciente a este dominio requieren de la experiencia y conocimientos de los expertos en esta área, en combinación con la experiencia y conocimientos del “minero” de datos, de manera que estos datos reflejen con precisión la naturaleza de dicho fenómeno.

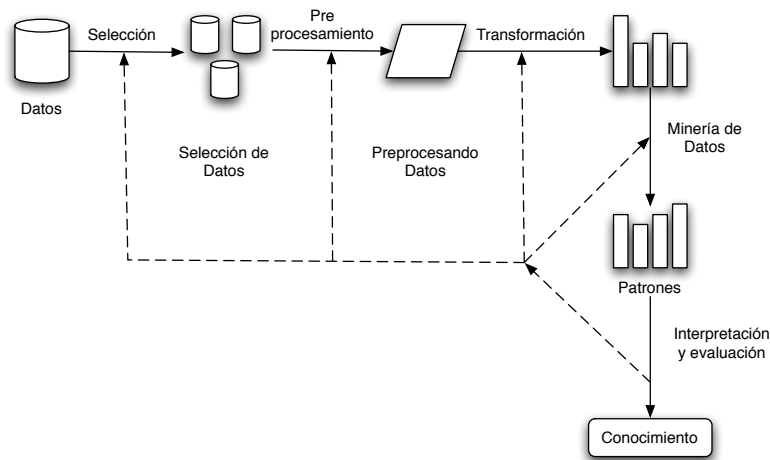


Figura 1. Proceso KDD.

La importancia del sistema Weka radica principalmente en que proporciona una gran diversidad de métodos (integrados en una amigable interfaz gráfica) para los pasos de KDD. Así, es posible analizar los datos usando diversos modelos y por lo tanto, formular nuevas hipótesis o bien, corroborar o refutar hipótesis existentes.

Para la selección de datos, Weka proporciona al usuario la facilidad de visualizar las características de las variables que conforman la base de datos original: tipo (nominal, numérica, etc.), distribución de probabilidad (histograma), valores máximo, mínimo, promedio y desviación estándar (para variables numéricas), entre otras. Además, Weka permite eliminar manualmente las variables que el usuario decida no incluir en el análisis así como ofrece métodos automáticos para la selección de atributos: ganancia en información, análisis de componentes principales, análisis de χ^2 , por mencionar algunos.

Dentro de las fases de pre-procesamiento y transformación, Weka pone a disposición algoritmos para la discretización de atributos numéricos, conversión de atributos nominales a binarios, normalización y llenado de valores faltantes, entre otros.

Para la etapa de minería de datos, Weka cuenta con algoritmos para aprendizaje supervisado (basados en reglas, árboles de decisión, clasificadores bayesianos, redes neuronales, entre muchos otros) así como algoritmos para aprendizaje no supervisado (basados en agrupaciones por cúmulos o “clusters” y en asociaciones).

Para la fase de evaluación e interpretación, Weka proporciona diferentes métodos: *hold-out*, validación cruzada (*k-fold cross-validation*), matriz de confusión, entropía, etc.

En resumen, Weka cuenta con diferentes herramientas para los distintos pasos del proceso KDD; todas ellas integradas en una interfaz gráfica que permite conectar las distintas fases y así, en el mejor de los casos, seguir el proceso KDD de principio a fin y lograr la meta última: conformar nuevo conocimiento en un dominio en particular.

4. Un Sistema Multi-agentes sobre Weka

¿Cómo podemos ofrecer soporte a un usuario que enfrenta las múltiples opciones que Weka ofrece en el proceso KDD? La figura 2 muestra una primera exploración del SMA que nos hemos propuesto con este fin. El SMA está conformado por cinco agentes, cuyos roles son como sigue: *Interface* actúa como coordinador dentro del SMA. Si las creencias de este agente, llegan a incluir *archivo(Path)*, donde *Path* se refiere a la ubicación de los datos, *Interface* pedirá al agente *Loader* que cargue la base de datos señalada por *Path* en un formato entendible para Weka (Orden de conversión). *Interface* comunica entonces a los agentes *NB* (Naive Bayes) e *ID3* un Orden de aprendizaje. *ID3* puede solicitar a *Discretize* que pre-procese la base de datos. *NB* puede acceder también a la base de datos discretizada. Los agentes *NB* e *ID3* comunican los modelos obtenidos al agente *Interface*, quien evalúa el mejor modelo y reporta al ganador.

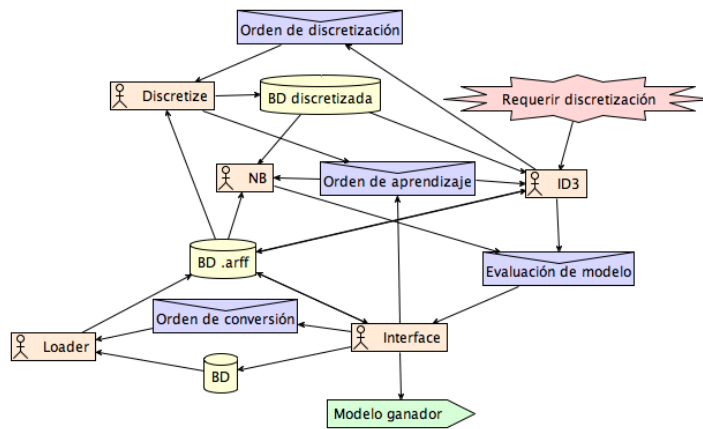


Figura 2. Diagrama del Sistema Multi-Agente.

4.1. Implementación

Como mencionamos, el SMA es modelado usando PDT. La implementación se lleva a cabo en Jason. Los algoritmos que provee Weka, pueden encapsularse como acciones primitivas de los agentes del SMA, gracias a que ambos sistemas están implementados en Java. Se han definido cinco acciones primitivas para este prototipo:

- **CargarArchivo:** Usada por el agente *Loader* para cargar, y convertir la base de datos (BD) al formato .arff si fue proporcionada en otros formatos (csv o xls). Recibe como argumento el path a la BD y devuelve la BD en format arff.

- *Discretiza*: Empleada por el agente *Discretize* para discretizar la BD. Recibe como entrada la BD en formato arff y devuelve la BD discretizada.
- *EvaluarModelo*: Usada por el agente *NB* para aprender un clasificador bayesiano usando Naive Bayes. Puede recibir la BD en formato arff o la BD discretizada, y devuelve el porcentaje de ejemplos clasificados correctamente.
- *EvaluarModeloID3*: Usada por el agente *ID3* para aprender el clasificador árbol de decisión usando ID3. Puede recibir la BD en formato arff o la BD discretizada, y devuelve el porcentaje de ejemplos clasificados correctamente.

En seguida se muestran algunos ejemplos del uso de estas acciones en los planes de los agentes del SMA:

- El plan *@cargaArchivo* del agente *Loader* ejecuta la acción primitiva *weka.cargarArchivo*. Como podemos ver, los argumentos de entrada y de salida de esta acción son: la variable *Path* que se refiere al camino a la BD e *Data* que es la BD en formato arff.

```
@cargaArchivo
+!cargarArchivo(X): true
  <- .print("Agente Loader cargando base de datos...");
     weka.cargarArchivo(Path,Data);
     .print(Data).
```

- El plan *@discretiza* del agente *Discretize* ejecuta la acción primitiva *weka.discretiza*. Los argumentos de entrada y de salida de esta acción son: la variable *Data* que se refiere a la BD en formato arff e *DataDiscr* que es la BD discretizada.

```
@discretiza[atomic]
+!discretiza(Data)[source(Ag)]: true
  <- .print("Discretizando BD...");
     weka.discretiza(Data,DataDiscr);
     .send(Ag, achieve, aprenderModelo(DataDiscr)).
```

- El plan *@naiveBayes* del agente *NB* ejecuta la acción *weka.evaluarModelo*. Sus argumentos de entrada y de salida son: la variable *Data* que se refiere a la BD en formato arff o a la BD discretizada y *Ev* que es porcentaje de clasificación del modelo Naive Bayes (evaluación).

```
@naiveBayes[atomic]
+!aprenderModelo(Data): true
  <- .print("Aprendiendo modelo Naive Bayes...");
     weka.evaluarModelo(X,N);
     .send(interface, tell, eval(nb,N)).
```

En total se han implementado una decena de planes que especifican las interacciones que se muestran en la figura 2. *ID3* es el agente que más planes tiene por el momento (cinco), incluyendo los planes para saber qué hacer si los datos no le son útiles, por ejemplo, si son continuos. Como puede observarse, los planes también

hacen uso de acciones primitivas (*.send*) de comunicación, especificadas como actos de habla en KQML.

5. Pruebas y Resultados

Por el momento, nuestros experimentos se han limitado reportar el modelo con mayor porcentaje de clasificación correcta, encontrado por el SMA. La siguiente tabla muestra algunos de los resultados que hemos obtenido con diferentes bases de datos. La cuarta y quinta columnas muestran el porcentaje de ejemplos clasificados correctamente por los modelos Naive Bayes e ID3, respectivamente. En todos los casos, nuestro SMA reporta correctamente el modelo ganador.

| BD | Atributos | Ejemplos | % NB | % ID3 |
|-------------------|-----------|----------|-------|-------|
| Iris | 4 | 150 | 94.67 | 94.00 |
| Contact-Lenses | 4 | 24 | 75.00 | 75.00 |
| Segment-challenge | 19 | 1500 | 91.27 | 94.47 |
| Segment-test | 19 | 810 | 90.74 | 92.10 |

Aunque estos experimentos son aún extremadamente simples, permiten evaluar la viabilidad de definir agentes BDI que atiendan el proceso KDD, en sus diferentes etapas. El SMA actual, lleva a cabo partes del preprocesamiento, transformación y minería de datos. La estrategia de meta-aprendizaje por votación, es la más sencilla de todas. En el otro extremo tenemos la combinación de modelos.

6. Conclusiones

Es sabido que las actuales herramientas de KDD incluyen una gran cantidad de algoritmos, pero carecen de lineamientos claros sobre el orden en que deben ser utilizados y su lo adecuado de una selección, de acuerdo a la naturaleza del problema estudiado. Como parte de los esfuerzos para atacar este problema, se han propuesto diversos SMA que lleven a cabo meta-aprendizaje en el proceso KDD.

Nosotros proponemos un marco de trabajo basado en Weka, una conocida herramienta de KDD, y Jason, un lenguaje de programación de agentes racionales BDI. Hemos implementado algunos prototipos de SMA para estudiar la viabilidad del enfoque. El sistema completo será desarrollado como parte de la tesis de maestría de Rosibelda Mondragón. Aún en esta etapa preliminar, es posible concluir que:

- La metodología Prometheus y Jason proveen un marco de trabajo lo suficientemente maduro, como para resolver el problema de implementar un SMA de soporte a los procesos KDD en Weka. Esta opción no es la única, como mencionamos en la sección 2. Lo relevante aquí es que independientemente de las herramientas de modelado y programación, como base se cuenta con el concepto de agente BDI.

- La programación de los agentes en términos de creencias, metas, intenciones, planes y eventos, facilita la comunicación con los expertos del área. Lo cual sin duda es el objetivo de la actitud intencional. Los agentes racionales proveen el nivel de abstracción adecuado para programar procesos KDD.
- La elección de Jason y Weka resulta extremadamente afortunada. Los agentes cuentan con una fuente importante de acciones primitivas. Esto nos permite concentrarnos en el problema de KDD, obviando el problema del aprendizaje automático. Lo cual nos lleva nuevamente a situarnos al nivel de abstracción adecuado.

El trabajo futuro incluye mejorar las competencias de los agentes del SMA, mejorar los protocolos de cooperación con base en técnicas de meta-aprendizaje; decidir que tipo de interacción se tiene con el usuario (supeditada a Weka o al SMA).

Referencias

1. S. Agüera, A. Guerra, and M. Martínez. Memory based reasoning and agents adaptive behavior. In O. Cairo, L.E. Sucar, and F.J. Cantu, editors, *MICAI 2000: Advances in Artificial Intelligence, Mexican International Conference on Artificial Intelligence, Acapulco, Mexico, April 11-14, 2000, Proceedings*, volume 1793 of *Lecture Notes in Computer Science*, pages 610–620, Heidelberg, Germany, 2000. Springer Verlag.
2. R. Bordini, M. Dastani, J. Dix, and A. El Fallah-Seghrouchni, editors. *Programming Multi-Agent Systems, Third International Workshop, ProMAS 2005, Utrecht, The Netherlands, July 26, 2005, Revised and Invited Papers*, volume 3862 of *Lecture Notes in Computer Science*. Springer Verlag, 2005.
3. R. H. Bordini and J. F. Hübner. Bdi agent programming in agentspeak using jason. In F. Toni and P. Torroni, editors, *Proceedings of the Sixth International Workshop on Computational Logic in Multi-Agent Systems (CLIMA VI), London, UK, 27-29 June, 2005, Revised Selected and Invited Papers*, volume 3900 of *Lecture Notes in Computer Science*, pages 143–164, Berlin, 2005. Springer-Verlag.
4. M.E. Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA., USA, and London, England, 1987.
5. M.E. Bratman, M.E. Pollak, and Israel D.J. Plans and resource-bounded practical reasoning. *Computer Intelligence*, 4:349–355, 1988.
6. D.C. Dennett. *The Intentional Stance*. MIT Press, Cambridge, MA., USA, 1987.
7. M. d’Inverno, D. Kinny, M. Luck, and M. Wooldridge. A formal specification of dmars. In M.P. Singh, A.S. Rao, and M. Wooldridge, editors, *Intelligent Agents IV: Proceedings of the Fourth International Workshop on Agent Theories, Architectures, and Languages*, volume 1365 of *Lecture Notes in Artificial Intelligence*, pages 155–176, Berlin-Heidelberg, Germany, 1998. Springer Verlag.
8. R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley & Sons, Inc., Danvers, MA., USA, 2001.
9. N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.
10. M.P. Georgeff and A.L. Lansky. Reactive reasoning and planning. In *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-87)*, pages 667–682, Seattle, WA., USA, 1987.

11. J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, San Francisco, CA., USA, 2001.
12. M. Huber. A BDI-theoretic mobile agent architecture. In *Proceedings of the Third Conference on Autonomous Agents (Agents99)*, pages 236–243, Seattle, WA., USA, 1999.
13. M. Huns and M.P. Singh, editors. *Readings in Agents*. Morgan Kaufman Publisher, San Mateo, CA., USA, 1998.
14. D. Jensen, Y. Dong, B. S. Lerner, E. K. McCall, L. J. Osterweil, S. M. Sutton, and A. Wise. Coordinating agent activities in knowledge discovery processes. In *Proceedings of the International Joint Conference on Work Activities Coordination and Collaboration*, San Francisco, California, 1999.
15. Matthias Klusch, Stefano Lodi, and Gianluca Moro. Issues of agent-based distributed data mining. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 1034–1035, New York, NY, USA, 2003. ACM Press.
16. B. Laurel, editor. *The Art of Human-Computer Interface Design*. Addison-Wesley Publishing Company, Inc., USA, 1990.
17. P. Luo, Q. He, R. Huang, F. Lin, and Z. Shi. Execution engine of meta-learning system for kdd in multi-agent environment. In *AIS-ADM*, pages 149–160, 2005.
18. Á. F. Moreira and R. Bordini. An operational semantics for a bdi agent-oriented programming language. In *Proceedings of the Workshop on Logics for Agent-Based Systems (LABS-2002) held with KR2000, April 22–25, Toulouse, France*, pages 45–59, 2002.
19. Lin Padgham, John Thangarajah, and Michael Winikoff. Tool support for agent development using the prometheus methodology. In *Fifth International Conference on Quality Software (QSIC 2005), 19-20 September 2005, Melbourne, Australia*, pages 383–388. IEEE Computer Society, 2005.
20. John Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
21. A.S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In Rudy van Hoe, editor, *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Eindhoven, The Netherlands, 1996.
22. A.S. Rao and M.P. Georgeff. Formal models and decision procedures for multi-agent systems. Technical Report 61, Australian Artificial Intelligence Institute, Carlton, Victoria, June 1995.
23. M.P. Singh, A.S. Rao, and M.P. Georgeff. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, chapter Formal Methods in DAI: Logic-Based Representation and Reasoning, pages 331–376. MIT Press, Cambridge, MA., USA, 1999.
24. SPSS. <http://www.spss.com/clementine/>.
25. R. Vilalta, C. Giraud-Carrier, and P. Brazdil. *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers.*, chapter Meta-Learning: Concepts and Techniques. Springer Publishers, 2005.
26. G. Weiß, editor. *Multiagent Systems, a modern approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA., USA, 1999.
27. I. H. Witten and E. Frank. *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, CA., USA, second edition, 2005.
28. M. Wooldridge. *Reasoning about Rational Agents*. MIT Press, Cambridge, MA., USA, 2000.
29. N. Zhong, Y. Matsui, T. Okuno, and C. Liu. Framework of a multi-agent kdd system. In *IDEAL*, pages 337–346, 2002.