Docencia de prácticas de Inteligencia Artificial en la Universidad de Alicante

Miguel A. Cazorla Quevedo, Otto Colomina Pardo y Rosana Satorre Cuerda

Departamento de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
{miguel,otto,rosana}@dccia.ua.es

Resumen

En este trabajo pretendemos abordar el enlace entre los contenidos teóricos que se tratan en las asignaturas troncales de I.A. y las sesiones de laboratorio. En este sentido optamos por el planteamiento de ejercicios prácticos que, tomando como referencia distintos lenguajes y herramientas, capaciten al estudiante para comprender las problemáticas básicas.

1 Introducción

En la Universidad de Alicante se imparten las asignaturas relacionadas con la Inteligencia Artificial (I.A.) que se muestran en la Tabla 1. Como puede observarse en dicha tabla hay un núcleo básico de asignaturas dedicado a los fundamentos de la I.A. y un conjunto de asignaturas específicas, dedicadas a la profundización en el estudio de problemáticas concretas o de técnicas avanzadas.

Tabla 1: Asignaturas de I.A. en la Universidad de Alicante

Asignatura	Curso	Carácter
Fundamentos de I.A.	4º	Troncal
Técnicas de I.A.	4º	Troncal
Ampliación de I.A.	-	Optativa
Aprendizaje	-	Optativa
Reconocimiento de formas	-	Optativa
Sistemas conexionistas	-	Optativa
Razonamiento geométrico	-	Optativa
Ampliación de lógica	-	Optativa

En este artículo se pretende exponer los contenidos prácticos de las asignaturas troncales: *Fundamentos de I.A. y Técnicas de I.A.* Estas dos asignaturas sientan las bases de los conceptos teóricos y prácticos que posteriormente se desarrollarán en las asignaturas más específicas. Por ello, el discente debe adquirir un buen nivel de conocimiento teórico y se pretende afianzar dicho conocimiento con los contenidos prácticos de ambas asignaturas.

Estas dos asignaturas, Fundamentos y Técnicas de I.A., tienen un volumen de créditos de 4.5, de los cuales 1.5 corresponden a contenido práctico. Dada la limitación temporal de las prácticas de ambas asignaturas, nos parece razonables orientarlos a reforzar los conceptos previamente vistos en las clases teóricas. Por ello, se busca que el discente profundice en los conceptos partiendo de implementaciones previamente suministradas. En cada bloque de la asignatura se distribuye un código que implementa algún concepto teórico. Sobre dicho código el alumno realizará una serie de pruebas de ejecución para determinar el funcionamiento (traza del algoritmo, flujo de datos, etc.) y se propondrán ligeras modificaciones de dicho código que permiten al alumno comprender la complejidad del problema tratado.

El artículo está organizado de la siguiente manera: en la Sección 2 se detalla el contenido teórico de las asignaturas; en la Sección 3 se comentan las herramientas utilizadas en las prácticas y en la Sección 4 se detallan los bloques de las asignaturas y el contenido básico desarrollado. Por último comentamos los resultados obtenidos en el desarrollo de este enfoque, así como su aceptación por parte de los alumnos. Para una mayor profundización de

las herramientas y ejemplos se puede consultar [2].

2 Contenido de las asignaturas

El contenido de las dos asignaturas a tratar se divide en los bloques temáticos que aparecen en la Tabla 2. En la asignatura de Fundamentos de I.A. se abordan una serie de problemas básicos para esta disciplina, como puede ser la búsqueda y la representación del conocimiento. En Técnicas de I.A. ([1]) se tratan áreas de aplicación más específicas, como puede ser visión artificial, aprendizaje o lenguaje natural. En la primera se profundiza más en los contenidos, ya que sienta las bases de conocimiento en el área, mientras que en Técnicas de I.A. esta profundización es menor y se presenta unos contenidos más descriptivos, no por ello menos rigurosos.

3 Herramientas utilizadas

Las prácticas de I.A. se desarrollan en un entorno hardware formado por PCs que funcionan como estaciones de trabajo bajo el sistema operativo Linux. La elección de las herramientas software a utilizar se ha realizado en base a dos criterios: en primer lugar, la preferencia por entornos abiertos que permitan al usuario realizar de manera sencilla sus propias modificaciones y aportaciones, y por otro lado la utilización de programas de libre distribución que hacen posible que el discente pueda utilizar libremente las mismas herramientas que se emplean en el aula si dispone de los medios adecuados.

En la Tabla 3 se muestra un resumen de las herramientas utilizadas en las prácticas y su relación con los bloques temáticos expuestos anteriormente.

3.1 LISP

Lisp, permite modelizar procesos de razonamiento. Son muchos los dialectos de Lisp existentes. Nos hemos basado en xLISP, debido a su portabilidad entre distintas máquinas y a su carácter de libre distribución. Para profundizar en Lisp se puede consultar el libro [7].

La conducta inteligente puede modelizarse con programas Lisp (paradigma simbólico de la I.A.). Lisp presenta algunas ventajas respecto a otros lenguajes:

Interacción Lenguaje interactivo, interpretado. Consulta y actualización de datos y procedimientos a voluntad.

Disponibilidad de entornos De uso extendido, sobre todo como soporte de aplicaciones de IA, incluyendo herramientas de edición y depuración.

Características propias Lisp fue diseñado desde un principio para manejo de símbolos. Por tanto reúne mejores características que otros lenguajes, sobre todo en versiones con instrumentaciones optimizadas que además incluyen características de otros lenguajes.

Uniformidad Las funciones y los datos tienen la misma forma. Un programa Lisp es un conjunto de listas por lo que otro programa Lisp puede usarlo como dato o bien crearlo y utilizarlo. Esto simplifica la sintaxis y facilita su aprendizaje.

Notación prefija La notación prefija es un tópico de Lisp. En esta notación, primero aparece el operador y luego los operandos. Si queremos sumar 4 y 5 en Lisp se realiza con `(+ 4 5)`.

Pero también existen algunas deficiciencias:

Ineficiencia Por ello ha sido tradicionalmente un lenguaje de prototipado, primándose la capacidad de interacción sobre la rapidez de ejecución de los programas.

Difícil lectura Los programas Lisp, usualmente extensos, incluyen multitud de paréntesis, consecuencia de la composición funcional, lo cual dificulta su lectura. Esto se puede solucionar con indentación adecuada y con facilidades de entorno (encontrar el paréntesis asociado a uno dado: editores tipo EMACS, ALPHA, VI).

3.2 CLIPS y FuzzyCLIPS

CLIPS es una herramienta de procesado simbólico utilizada, fundamentalmente, para el

Tabla 2: Contenido de las asignaturas

Fundamentos de I.A.	Técnicas de I.A.
Búsqueda heurística	Lenguaje natural
Propagación de restricciones	Visión artificial
Búsqueda en juegos	Lógica difusa
Representación del conocimiento	Aprendizaje

Tabla 3: Herramientas software utilizadas

Herramienta	Bloque temático	Conocimientos desarrollados
LISP	Búsqueda heurística Propagación de restricciones Búsqueda en juegos	Algoritmo A^*, A^*_ϵ Algoritmo de Waltz Minimax, $\alpha - \beta$, SSS^*
CLIPS	Lenguaje natural Representación del conocimiento	Análisis morfológico y sintáctico Sistemas expertos
FuzzyCLIPS	Lógica difusa	Sistemas expertos difusos
VISTA	Visión artificial	Algoritmos de esqueletización y obtención de características
SNNS	Aprendizaje	Entrenamiento de redes neuronales

desarrollo de sistemas expertos y en general en proyectos de Inteligencia Artificial. Fue desarrollado en NASA en 1985 y es posible encontrar variedad de aplicaciones desarrolladas en este lenguaje. Las siglas CLIPS significan: C Language Integrated Production System (Sistema de producción integrado en lenguaje C). Como su nombre indica, CLIPS esta desarrollada en C, por lo que presenta una ventaja a la hora de integrar nuevos módulos.

Tres son las formas para representar el conocimiento en CLIPS:

- Reglas. Conocimiento heurístico basado en experiencia.
- Funciones genéricas. Conocimiento procedural.
- Programación orientada a objetos.

CLIPS nos proporciona los elementos básicos de un sistema experto. Estos son: Base de hechos, base de conocimiento y el motor de inferencia. Un programa en CLIPS deberá contener los hechos y las reglas de producción de nuestro sistema. El motor de inferencia viene incorporado en CLIPS y va a permitir elegir entre una serie de estrategias ya implementadas (profundidad, anchura, etc.). Un programa en CLIPS consiste en definir un conjunto de reglas para resolver el problema. CLIPS detecta cuando se cumple alguna de estas reglas y la ejecuta, pudiendo modificar la base de hechos.

FuzzyCLIPS [3] es una extensión de CLIPS, siguiendo la misma sintaxis, pero como soporte para los sistemas expertos difusos. El sistema integra de forma ponderada los consecuentes para dar una respuesta global afectada por la confianza o nivel de disparo de cada una de las reglas.

3.3 VISTA

La librería Vista [5] es un conjunto de aplicaciones, rutinas y algoritmos orientados al tratamiento de imágenes por ordenador. Se empezó a desarrollar en la Universidad de British Columbia de Vancouver (Canadá), pero se ha ido ampliando con aportaciones de los usuarios. Se viene utilizando en un gran número

de universidades e institutos tecnológicos, de donde cabe destacar el MIT (Massachuset Institute of Technology) o Carnegie Mellon University. Vista es un software de libre distribución multiplataforma, por lo que la hace muy atractiva en temas relacionados con la docencia.

La filosofía de Vista no busca la disponibilidad de un gran entorno gráfico de desarrollo, sino un sistema potente donde la incorporación de nuevos tipos, adición de nuevas rutinas y programas sea lo más sencilla posible y el usuario no deba de preocuparse por temas tales como manejo de X Windows, funciones específicas de la plataforma donde estamos instalando Vista, etc. Esta es una de las principales ventajas en Vista: permite la instalación y utilización en diversas estaciones de trabajo sin tener que modificar código. Para ello se han desarrollado todas las rutinas en C estándar y las utilidades gráficas en X Windows. Esto lo hace muy recomendable para la enseñanza, pues ya no es necesario grandes plataformas, sino que se puede trabajar en ordenadores personales sin mayor problema.

3.4 SNNS

El SNNS (Stuttgart Neural Network Simulator) es un paquete software que permite la construcción, simulación y entrenamiento de diversos modelos de Redes Neuronales Artificiales. El sistema incorpora un entorno gráfico que permite ajustar interactivamente y de manera intuitiva un gran número de parámetros. Permite el desarrollo de prototipos que, una vez comprobados, pueden ser convertidos a código C para su utilización en un sistema externa.

4 Bloques temáticos

En cada bloque temático se propone la resolución de un problema que le sirva al alumno como introducción a las técnicas apropiadas y a la vez como motivación para su estudio.

A continuación se exponen, de manera más detallada, algunos de los objetivos a alcanzar con cada bloque temático y las herramientas utilizadas, así como algunos ejemplos de problemas y ejercicios que se les plantean a los alumnos para afianzar los conocimientos adquiridos.

4.1 Búsqueda heurística

Como ejemplo de aplicación de la búsqueda heurística se propone el problema de la correspondencia entre contornos. Dado un conjunto de contornos planos (conjunto de test) y un contorno objetivo, se trata de encontrar el contorno de test de mayor correspondencia (parecido) con el objetivo. La Figura 1 refleja el problema a tratar.

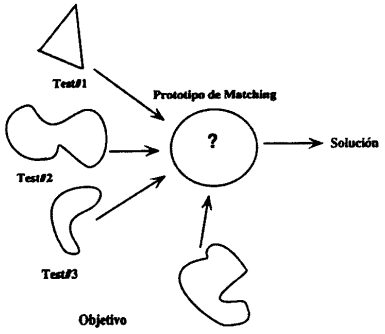

Figura 1: El problema se define como averiguar cuál de las formas de test se corresponde con la objetivo.

Para resolver este problema se utiliza una implementación en LISP del algoritmo A^* [4]. El alumno debe ir analizando cada una de las partes del problema: definición del espacio de estados y las reglas de producción, formulación de las funciones $g(x)$ y $h(x)$. El plan de trabajo consiste en que el alumno realice varias ejecuciones del algoritmo, realizando un seguimiento exhaustivo de cada paso del algoritmo. Una vez realizada esta parte, se propone que se amplíe el código para ejecutar el algoritmo A^*_ϵ.

4.2 Propagación de restricciones

Vamos a considerar el problema de etiquetar figuras para un posterior reconocimiento. Si tenemos una vista de la figura, las líneas de esta vista representan al objeto en dos dimensiones. Dado un punto de unión de dos o más aristas, las etiquetas que se pueden aplicar a

las aristas que convergen a dicho punto es variada, pero algunas de ellas son físicamente imposibles, de tal manera que nos restringen el número de etiquetas a aplicar.

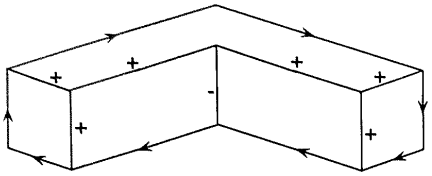

Figura 2: Etiquetado de una figura con vértices de tres caras.

En este bloque se suministra también un código LISP que implementa el algoritmo de Waltz [6] de etiquetado de aristas. Este algoritmo, dado un conjunto de vértices y aristas realiza un etiquetado como el de la Figura 2, aplicando satisfacción de restricciones. El alumno debe realizar una traza del algoritmo para, posteriormente, ampliar el conjunto de etiquetas para que pueda manejar figuras con sombras y vértices de más de tres aristas.

4.3 Búsqueda en juegos

El ajedrez es un juego de estrategia muy completo y de gran aceptación que proporciona un amplio campo de trabajo para el estudio de métodos complejos de resolución de problemas. Con este problema pretendemos abordar, a partir de una instrumentación básica en lenguaje Lisp del juego, la construcción de un sistema de producción completo que juegue al ajedrez en base a distintas estrategias de control o de búsqueda en juegos, así como la evaluación de la eficiencia de dichas estrategias.

El alumno debe incorporar al código básico, suministrado con MINIMAX, alguna estrategia de búsqueda (alpha-beta, SSS^*, etc.) para comprobar la mejora que se produce al utilizar dichas estrategias.

4.4 Aplicación al procesamiento de Lenguaje natural

En este bloque se pretende profundizar en el procesamiento de lenguaje natural, abordan sus problemáticas tempranas como son el procesamiento morfológico y el sintáctico. La implementación suministrada se encuentra desarrollada en lenguaje CLIPS. El algoritmo de análisis sintáctico utilizado es el de análisis ascendente, de muy sencilla implementación en CLIPS. Aquí el alumno realiza un análisis del flujo de datos del algoritmo, que permite una mayor asimilación del algoritmo visto en teoría.

4.5 Sistemas expertos y lógica difusa

Se aborda el problema del aparcamiento automático, en el que se pretende aparcar un móvil en una determinada posición con una determinada orientación (Figura 3) mediante un sistema experto difuso.

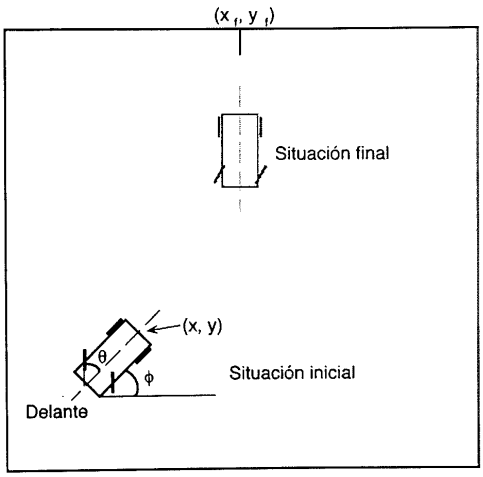

Figura 3: Aparcamiento de un móvil.

Se parte de un sistema de reglas clásico definidas en CLIPS que resuelven este problema. El alumno debe resolver el mismo problema con lógica difusa, utilizando la herramienta FuzzyCLIPS. Una vez desarrollado el sistema difuso se procede a una comparación de resultados entre los dos sistemas que lleva al alumno a interesantes conclusiones sobre la forma de funcionamiento de un sistema difuso y las ventajas de utilizar este tipo de sistemas. Además, con los conocimientos adquiridos en este bloque el alumno sienta las bases de la asignatura Ampliación de lógica.

4.6 Aplicación a la visión

Para introducir al alumno en la problemática de la visión de nivel bajo y medio se realizan dos tipos de prácticas utilizando la librería VISTA.

En un primer nivel se aborda el uso de los diversos programas que proporciona VISTA para el tratamiento de imágenes, para lo que se propone al alumno la realización de una serie de ejercicios de aplicación de filtros de convolución, concatenación de imágenes, cálculo de aristas, etc. Al disponer en VISTA de una implementación de los algoritmos de visión que se han tratado en las clases teóricas, el alumno puede experimentar con los mismos y de este modo alcanzar una comprensión más profunda de su funcionamiento y aplicaciones.

En un segundo nivel se utiliza la posibilidad de programar utilizando las rutinas de la librería para la resolución de un problema más concreto en el campo de la visión. En este caso, disponemos de dos ejemplos: uno obtiene esqueletos de imágenes binarias (Figura 4) y otro obtiene un vector de características a partir de imágenes de caracteres. Al alumno se le proporciona una implementación en VISTA de uno u otro algoritmo y debe comprobar su funcionamiento, ampliando el código en algunos casos.

Figura 4: Región binaria y el resultado de aplicar el algoritmo de esqueletización.

Este bloque posee una mayor dedicación debido a que sirve de enlace con las asignaturas optativas: Reconocimiento de formas, ampliación de I.A. y razonamiento geométrico

4.7 Aplicación al aprendizaje

Este software se aplica al problema de entrenamiento de un perceptrón multicapa que permita reconocer caracteres. Al alumno se le proporciona un fichero con los ejemplos de entrenamiento y se le pide que utilizando el SNNS construya la red neuronal y ajuste los parámetros de entrenamiento hasta conseguir un resultado satisfactorio.

El objetivo de esta práctica es que el alumno adquiera una comprensión básica de las capacidades y limitaciones de las redes neuronales y de las diversas problemáticas que puede plantear el proceso de ajuste. El alumno puede comprobar también las interrelaciones entre varios bloques de la asignatura.

Como ejercicios adicionales se proporcionan ejemplos de entrenamiento para resolver problemas de aprendizaje más complejos y se plantea la resolución de problemas que requieran la utilización de otros modelos de redes neuronales. Los conocimientos de este bloque se ven con más profundidad en la asignatura Sistemas conexionistas y Aprendizaje.

5 Conclusiones

Se ha presentado un enfoque de la docencia práctica de la Inteligencia Artificial basada en los siguientes puntos:

- Se utilizan diversas herramientas software especialmente adaptadas a los bloques temáticos en que se dividen las asignaturas. Se ha buscado siempre el uso de herramientas de libre distribución para que el alumno
- Al alumno se le suministra una implementación de los algoritmos básicos, lo que le permite concentrarse en la comprensión de los conceptos más que en los detalles de implementación.
- Se propone al alumno que amplíe el código suministrado para que conciba la complejidad en la implementación del problema.

Es de destacar la aceptación que este enfoque ha tenido en el alumnado de las asignaturas en las que se ha aplicado, habiendo conseguido los alumnos por un lado un alto grado de satisfacción en la realización de las prácticas y por otro un afianzamiento profundo de los conceptos teóricos y su aplicación práctica. Estos hechos se reflejan en el elevado índice de éxitos entre los alumnos que tienen las

asignaturas de I.A. pese a la complejidad de las materias que se tratan.

Referencias

[1] M. Cazorla, O. Colomina, F. Escolano, D. Gallardo, R. Rizo, and R. Satorre. *Técnicas de Inteligencia Artificial.* Editorial Club Universitario, 1997.

[2] M. Cazorla, F. Escolano, R. Rizo, and R. Satorre. *Laboratorio de Inteligencia Artificial.* Publicaciones de la Universidad de Alicante, 1998.

[3] R.A. Orchad. *FuzzyClips Version 6.04, User's guide.* Knowledge Systems Laboratory, Institute for Information Technology, National Research Council, Canada, 1995.

[4] J. Pearl. *Heuristics. Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, 1984.

[5] Arthur R. Pope and David G. Lowe. Vista: A software enviroment for computer vision research. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 1994.

[6] D. Waltz. Understanding line drawings of scenes with shadows. In P.H. Winston, editor, *The Psycology of Computer Vision.* Mc Graw Hill, 1975.

[7] P.H. Winston and B.K.P. Horn. *Lisp. Second Edition.* Addison-Wesley, 1984.