Docència de tècniques d'IA mitjançant Javasoccer

Josep Lluís de la Rosa **Miquel Montaner**
Departament d'Electrònica, Informàtica i Automàtica
Universitat de Girona
peplluis@eia.udg.es mmontane@eia.udg.es

Resum

Aquesta comunicació mostra una experiència docent desenvolupada mitjançant un simulador del joc de futbol. El joc de futbol potser serà la plataforma representativa de la nova IA. Aquí emprem una versió simplificada, el JavaSoccer, que ha permès aplicar de forma comparada tècniques ara estàndards de la IA dels 80 i 90: lògica difusa, sistemes experts, xarxes neuronals, algorismes genètics, agents i programació orientada a objectes, i es treballa l'embedded AI.

Mots clau: IA, Java, agents, JavaSoccer, SE.

1 Introducció: l'Assignatura

L'experiència docent s'ha desenvolupat amb una assignatura optativa de 5è d'Informàtica on els estudiants ja tenen una base d'IA per la troncalitat dels seus estudis. Han estat 20 estudiants. Dins del pla d'estudis l'assignatura conté 3 Crèdits de Teoria + 3 Crèdits de Pràctiques, i és accessible al campus virtual http://brakali.udg.es/~plang/ (com a usuari "convidat")

1.1 Objectius de l'Assignatura

Desenvolupar les habilitats en la IA més pràctica de la actualitat, vist des d'un punt de vista funcional i insistint en la inclusió de la IA en sistemes existents, usant per això un entorn Java que dóna eines naturals per implementar sistemes intel·ligents portables d'ús general. La introducció d'un factor lúdic i competitiu ha estimulat prou la qualitat dels desenvolupaments pels estudiants dins d'aquest curs.

1.2 Temari de l'Assignatura el Curs 1998/99

Tema 1: Introducció als Agents Intel·ligents
(2h) *Setmana 1*

- Concepte d'agent i sistema intel·ligent.
- Evolució històrica. Camps d'aplicació i exemples.

Tema 2: Enfoc pràctic de la lògica fuzzy (difusa)
(4h) *Setmanes 1 i 2*

- Incertesa i imprecisió
- Lògica binària i Lògica fuzzy: conjunts fuzzy.
- Aplicacions i exemples.

Pràctica 1 : Control fuzzy d'un viga-bola. (Matlab)

Tema 3: Enfoc pràctic de les xarxes neuronals
(4h) *Setmanes 3 i 4*

- Aprenentatge. Mecanismes.
- Aplicació i exemples.

Pràctica 2 : Xarxes Neuronals: Aplicacions (Matlab)

Tema 4: Enfoc pràctic dels algorismes genètics
(2h) *Setmanes 5*

- Adaptació i optimització.

Pràctica 3: Genètics (Matlab)

Tema 5: Enfoc pràctic dels sistemes experts
(6h) *Setmanes 6 i 7*

- Representació del coneixement
- Mecanismes d'inferència: Forward chainning and Backward chainning
- Introducció a JESS (Java Expert System Shell)
- Aplicacions i exemples

Pràctica 4: Repàs de Java i embedding JESS
Pràctica 5: Sistema expert JESS.

Tema 6: Agents Intel·ligents
(6h) *Setmanes 8, 9 i 10*

- Introducció. Agents software i agents físics.

- Consensus.
- Llenguatge AGENT0.
- Agents físics. Exemples.
- Disseny d'agents físics.

Pràctica 6: JavaSoccer
Pràctica 7: Disseny d'agents software i hardware.

Tema 7: Exemple d'aplicació a la supervisió i decisió de robots mòbils autònoms i cooperants (8h) *Setmanes 11, 12 i 13*

- Estudi de la problemàtica de la decisió reactiva.
- El problema de la decisió cooperativa.
- Aplicació de diferents paradigmes de resolució
- Avaluació comparada de resultats.

Pràctica 8 (Treball de curs) competició de futbol.

Material de Curs:

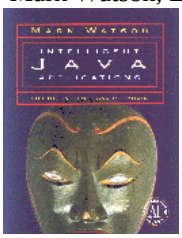
– JESS 4.1b3 o posterior.
– Compilador de Java JDK 1.1.x (o posterior)
– JavaSoccer 1.9b o posterior.
– MATLAB 4.2c i SIMULINK 1.3 i les seves Fuzzy 1.0 i neural 1.0 toolboxes (o posterior).
– Campus virtual IITAP http://eia.udg.es//~iitap

Pla Organitzatiu:

Comencem per 2 setmanes teoria (8 hores). Iniciem pràctiques a la 3a setmana amb 2 hores setmanals de pràctiques i 2 de teoria. Les 3 darreres setmanes són només pràctiques. L'avaluació es fa amb l'entrega dels treballs (grups de 2 persones), entrevista i competició.

Bibliografia i Recursos:

1. Intelligent Java Applications for the Internet and Intranets by Mark Watson, Ed. Add-Wesley.



2. Articles varis que es poden obtenir d'aquestes planes de web següents:
   http://www.robocup.org/research/56.html
   http://www.robocup.org/research/52.html
3. Java http://java.sun.com/products/jdk/1.1/index.html
4. Sistema Expert http://herzberg.ca.sandia.gov/JESS/
5. Javasoccer
   http://www.cc.gatech.edu/grads/b/Tucker.Balch/JavaBots/JavaSoccer/docs/

Treballs de Curs Proposats:

(1) Fuzzy 1. Implementació orientada a objectes.
(2) Fuzzy 2. Implementació neuronal.
(3) Neuronal 1. Aprenentatge off-line.
(4) Neuronal 2. Aprenentatge on-line
(5) Genètics 1. Aprenentatge off-line.
(6) Genètics 2. Aprenentatge on-line.
(7) Sistema expert JESS (explicat amb detall aquí).
(8) Sistema expert JESS 2. Implementació amb dispar de regla mitjançant lògica difusa.
(9) Agent + consensus 1. Normal.
(10) Agent + consensus 2. Implementació adaptativa.

Qualificació: s'assignaran aquestes tècniques fins a 10 grups de dues persones. Els grups amb temes 2, 4, 6, 8, 9 o 10 podran conseguir la màxima nota per mor que les tècniques adaptatives on-line són de major complexitat a part d'integrar diverses tècniques d'IA.

Els equips es construeixen sobre l'equip bàsic de la classe *class JavaSoccer.teams.BrianTeam.* Aleshores s'heretaran tots els recursos d'aquests equip bàsic.

2 Comparació de Tècniques d'IA

Aquest curs és una possible aproximació a l'ensenyament de la IA. I ara discutirem una mica més en detall sobre els aspectes filosòfics de per què plantejar un curs amb un entorn simulat de futbol, i per què comparar les tècniques d'IA d'aquesta manera (certament no estàndard).

2.1 Mancances de la Formació en IA

Una de les mancances de la formació en IA és el fet de només donar solucions completes que funcionen únicament en exemples hipersimplificats i acadèmics. La comunitat d'IA no és la única comunitat acadèmica que li succeix això doncs la comunitat d'Automàtica també pateix d'aquesta lacra: un 95% del treball sobre teories de control avançat només són aplicables en menys d'un 5% dels casos reals. En canvi un 5% del treball se desenvolupa sobre PID però s'aplica de forma generalitzada a la indústria. Més d'un pot trobar una exemplificació similar en IA on només qualques sistemes basats en lògica difusa i/oxarxes neuronals han trobat aplicacions més o menys generals i exteses.

Com a conseqüència els alumnes disposen d'un aparell teòric molt formatiu però que no el saben aplicar. Les experiències d'emprar IA (o control avançat en el cas d'automàtica) en la indústria són tímides i frustrants.

Una solució a això és dotar els alumnes de capacitat crítica suficient per sebre en quines condicions una tècnica d'IA és aplicable o és millor que un altra. És a dir, la docència li manca enfocs comparats de tècniques d'IA (d'altra banda, també la docència d'automàtica pateix del mateix...).

2.2 Mancances dels Resultats de Recerca en IA

L'any 1997 passarà a la història per ser l'any on per primer pic un ordinador va guanyar u-contra-u dins torneig oficial al campió mundial d'escacs Gary Kasparov. Els escacs es considerava com a un dels problemes paradigmàtics de la IA dels anys 50 i 60. Es pot entendre ara que som a la fi d'una època, la de resolució de problemes simbòlics (de joguina o *toy problems o toy examples* per qualques recercaires), i al començament d'una altra més propera als problemes del món real. Aquesta nova época en recerca d'IA implica plantejar nous reptes amb els que la IA pugui comprovar els seus avanços. Com que hi ha precedents satisfactoris de l'ús de jocs (els jocs permeten comparar propostes de recerca a més a més qu'alcen interés en les joves generacions i la societat) i tenint en compte que se han generalitzat les competicions robòtiques, en conseqüència un dels problemes proposats com a nou repte d'IA és el futbol robòtic.

D'altra banda, la puixant moda de recerca en agents dels anys 90 sembla qu'ha ben acceptat el futbol robòtic. La motivació d'experimentar la IA mitjançant robots es l'essència mateixa dels agents i agents físics. El fet d'utilizar robots per experimentar agents físics no és una decisió a l'atzar sinó d'un gran valor científic per cercar una refundació dels principis físics o biológics de la IA [Kitano 94, pp:34-35]. S'intenta descubrir els principis de la introspecció i aprenentatge introspectiu dels agents des del seu fonament físic. La construcció de sistemes intel·ligents i adaptatius no és una tasca trivial. Kitano prediu que l'aproximació al desenvolupament d'un sistema intel·ligent i fer-lo adaptatiu seria repetir els errors de la IA tradicional, la qual va suposar que la solució d'exemples simples (toy examples) es podria generalitzar a aplicacions de major complexitat, expectativa que no s'ha cumplert. Aleshores creiem que per aconseguir sistemes adaptatius intel·ligents, ells n'han de presentar les característiques següents: **emergència**, que vol dir obtenir intel·ligència emergent de les interaccions més simples entre components, **evolució**, la intel·ligència es fruit de l'evolució i millora continuada dels éssers intel·ligents, **simbiosi** entre diversos components heterogenis, diversitat per aconseguir una correcta simbiosi i afavorir l'emergència i l'evolució, motivació qui dirigeix la intel·ligència (com pot ser la motivació de l'aprenentatge) i **fonament físic**. Insistim en la importància del fonament físic per mor que la intel·ligència està governada per la física: els sistemes intel·ligents finalment es basen en un cos físic.

La idea de robots que jugassen al futbol fou proposada per primer pic pel professor Alan Mackworth de la universidad de British Columbia, (Canadà) el 1992 [Mackworth 93]. El projecte s'anomenà *Dynamo* [Sahota 95] i ha servit d'inspiració per a la *RoboCup*.

Indepenentment, un grup d'investigadors japonesos organitzaren un *Workshop* "Grand Challenge in Artificial Intelligence" a Tokio l'Octubre de 1992 per discutir possibles problemes que servissin com a test per a la IA. El juny de 1993 es proposà seriosament utilitzar el futbol per promocionar la ciència i la tecnologia amb un projecte denominat originalment J-League on hi participaven els recercaires Minoru Asada i Hiroaki Kitano. Per mor de l'interés contagiat fora de Japó es va decidir extendre-lo en forma de projecte internacional, que s'anomenà *Robot World Cup Initiative*, abreujat *RoboCup*. L' *ElectroTechnical Laboratory* (ETL), un centre de recerca japonés, va començar aleshores la recerca en sistemes multi-agents emprant el futbol, i engegà el desenvolupament d'un simulador de partits de futbol.

En la *National Conference on Artificial Intelligence* (AAAI-94) celebrada a Seattle (Estats Units) el 1994, es va fotre una discusió més detallada sobre la iniciativa *RoboCup*, amb investigadors dels Estats Units, Europa i Japó. El mateix any, l'ETL anunciava la primera versió en LISP del *Soccer Server*, el simulador oficial de la RoboCup.

Durant la *International Joint Conference on Artificial Intelligence* (IJCAI-95) a Montreal (Canadà) l'agost de 1995, s'anuncià la celebració dels primers **Robot World Cup Soccer Games and Conferences** que esclafiren a l'IJCAI-97 a Nagoya (Japó). Aleshores es va presentar la primera versió del JavaSoccer qu'emprem en aquest article.

2.3 Proposta: Fer Treballs Comparats en Entorns menys Estructurats. Insistir en la Embedded IA

El JavaSoccer és una versió reduïda implementada en Java del simulador oficial *Soccer Server*, desenvolupada en el Georgia Tech.

Per ser una versió reduïda facilita que els estudiants de 1r i/o 2n cicle d'una carrera universitària puguin aprendre amb facilitat el funcionament del simulador, consistent en objectes i mètodes Java, puguin incloure codi d'IA implementat en Java i fer un treball d'IA juntament amb una competició. El JavaSoccer dóna la facilitat suficient com per poder impartir el curs en les 15 (o 13!) setmanes lectives...

D'altra banda, encara que sigui una versió reduïda, el JavaSoccer conserva aspectes essencials del futbol: és un exemple poc estructurat on hi ha nombroses situacions que requereixen d'extreure tot el suc a les tècniques d'IA que hi apliquem. Representa l'essència de la nova IA consistent a desenvolupar codi basat en algorismes de la IA però que s'han d'integrar (embed) amb altres codis tant d'IA com convencionals. Finalment, el fet de programar un equip complet de 5 agents colaboratius – competitius dóna un ventall bastant complet dels nous reptes de la IA.

Aleshores resaltem que pels alumnes aquest és un tipus de treball on poden aplicar tècniques d'IA a un entorn concret i petit, tot i que a la vegada no és de joguina.

Finalment, no s'ha de passar per alt que la **competició** és molt necessària doncs, a part d'estimular fortament els estudiants a complir els terminis de l'assignatura, els facilita el necessari *enfoc comparat de les tècniques d'IA* aplicades pels altres grups de treball. De fet, durant la competició cada grup de treball exposa en públic els punts forts i febles de la seva aproximació i la resta d'alumnes els escolta amb molta atenció i prenen nota! Evidentment qu'els finalistes són els més escoltats i aleshores la tasca del professor és de relativitzar una mica els resultats obtesos per cada equip. Com a anècdota, en Bernat Casero, representant de l'equip que havia aplicat algorismes genètics amb aprenentatge on-line gemegava fort que la seva tècnica era massa lenta en convergir i que els seus jugadors eren extremadament lents a prendre decisions amb respecte els dels altres equips. Efectivament, va ser el seu equip sacsejat per 18-0 i 24-0. Emperò, la meva valoració academicista de la seva aportació era alta i el seu treball va ser qualificat com un 8,5. Quina contradicció segons els alumnes!

Sembla una contradicció valorar treballs que no "funcionen"? Efectivament, amb aquesta competició queda de relleu que hi ha tècniques molt ben valorades per la comunitat científica però, massa habitualment, no es coneix l'abast d'aplicabilitat de les esmentades tècniques. Afortunadament he pogut comprovar que amb l'impartició del curs d'aquesta manera els alumnes extreuen les seves pròpies conclusions d'una forma global i més independent de les modes qu'els professors/investigadors patim massa sovint. Almenys no s'en duran decepcions estrepitoses quan apliquin la IA a la seva carrera professional, amb el conseqüent risc d'avorrir la IA i enterrar-la per sempre, sinó que tindran una aproximació útil i realista.

La competició, amb 10 equips, es va desenvolupar durant 6 hores d'un matí el 18 de gener de 1999. Al partit final hi va anar, a part de la totalitat dels alumnes del curs, devers uns 50 alumnes de cursos inferiors així com una vintena de professors de l'Escola Politècnica Superior i un especialista en robòtica mòbil de la Universitat de Hawai, el Prof. Yunku Yuh. L'equip que va guanyar, **1ou7kou**, va aplicar algorismes de decisió basats en consensus i adaptació dels paràmetres de consensus, que tal com (ehem, ehem) el professor de l'assignatura esperava (l'autor d'aquest article que no pot evitar projectar la seva visió de futur) varen ser els de millor rendiment...

De tota manera, el tema de mètodes sistemàtics de comparació de les tècniques d'IA amb la plataforma d'emulació del joc de futbol encara no esta ben desenvolupat i és un tema que treballarem els propers cursos.

3 Un exemple: el Sistema Expert JESS

3.1 JESS: Java Expert System Shell

Dels treballs de curs presentats mostrarem aquí les possibilitats d'usar i aprendre el sistema expert JESS. És un sistema expert clònic de CLIPS que funciona en Java i que l'hem emprat integrant el motor d'inferència Rete de JESS dins del JavaSoccer. JESS s'ha emprat per dissenyar la decisió dels sistemes intel·ligents. El JavaSoccer i el JESS són força senzill en concepte i per tant tenen molt bona assimilació pels estudiants, i permeten amb certa facilitat treballar amb múltiples aspectes del qu'hauria de ser l'aplicació de la IA: representació del coneixement, anàlisi, recopilació i codificació de coneixement, i integració de la decisió experta dins d'un sistema (en aquest cas el JavaSoccer).

Volem fer un SE en jugar a futbol de forma que prengui en cada situació una decisió encertada. El primer que necessitem és el coneixement, per tant hem d'acudir a un expert en futbol per tal que ens expliqui tot el que sap però en forma de regles. Per exemple:

<< Tinc la pilota i no em marquen llavors avançar >>

La representació de coneixement sembla senzilla, però com en tot procés de representació de coneixement prest l'alumne veu que no és així. La majoria d'ocasions és molt difícil que l'expert sàpiga expressar-nos el seu coneixement amb un conjunt de regles. Les coses, realment, no solen ser tan lògiques perquè això sigui possible, i molts cops les nostres decisions tenen part d'intuïció i subjectivitat, entre d'altres. Bé, es diu que de futbol tothom hi entén, no? Encara així i tot la solució no és immediata.

3.2 JESS dins el JavaSoccer: Integració d'IA

El següent aspecte que cal comentar és l'associació que realitzem entre el JavaSoccer i JESS per treballar amb tots dos a la vegada. El primer que hem de dir és que la comunicació entre ambdós és realment senzilla, i des de Java es pot treballar molt bé amb el JESS. Bàsicament hi ha dos aspectes importants en aquesta comunicació: la inicialització i el traspàs d'informació.

Inicialització:

Per tal d'executar un fitxer de JESS (amb extensió *.clp*) des de Java tan sols hem de crear un objecte JESS el qual té com a paràmetres: el fitxer de JESS com a entrada i la pantalla com a sortida de dades, i un objecte *Rete*. Aquest objecte és l'enllaç que des de Java ens permetrà la interacció amb el JESS.

També és important no oblidar-nos d'incloure en el programa Java: *import JESS.** D'aquesta forma aconseguim que el programa en Java pugui utilitzar les funcions de JESS. Veiem com realitzem aquesta inicialització:

```
if (inici == true)
{           do  { parser.parse(false);}
            while (clp.available()>0);
            inici = false;
}
```

Tan sols ens falta el darrer pas per acabar la inicialització i consisteix en executar un constructor per a l'objecte JESS. Fixem-nos en com ho faríem:

El fet de construir l'objecte tan sols s'ha de realitzar una vegada i per això incloem una variable *inici* la qual és certa a l'inici del programa.

Per tal d'executar el constructor de l'objecte JESS tan sols hem d'aplicar el mètode *parse* sobre el nostre objecte. Fixem-nos, però, per tal que es construeixi correctament hem d'incloure'l dins un bucle que es va repetint fins que realment la construcció s'ha realitzat.

```
FileInputStream clp;              // Fitxer escrit en JESS
Rete JESS_engine;                 // Objecte Rete
Jesp parser;                      // Objecte JESS
NullDisplay nd;                   // Pantalla
public void Configure()
{           try
            {// Indiquem el fitxer clp escrit en JESS
               clp = new FileInputStream(NOMFITXER);
             // La sortida de dades del JESS es farà a pantalla
               nd = new NullDisplay();
             // Creem un objecte Rete
               JESS_engine = new Rete(nd);
             // Creem l'objecte JESS
               parser = new Jesp(clp, JESS_engine);
            }
            // Resposta a les excepcions
            catch (FileNotFoundException fnf)
            { System.out.println("error fitxer no trobat"); }
            catch (IOException ioe)
            { System.out.println("error excepcio"); }
 }
```

Comunicació:

Dins el nostre programa Java, i un cop creat l'objecte JESS, interessa interactuar amb el fitxer de JESS per realitzar tres accions bàsiques: 1r, enviar dades, 2n, indicar-li que s'executi i 3r, rebre els resultats.

El primer aspecte és l'enviament de dades que es fa mitjançant la funció *store*. La seva sintaxi és:

Public Value store(string name, Value val);

Un exemple podria ser:

JESS_engine.store("pilota_gol", new Value (pilota_gol.r, RU.FLOAT));

Amb aquesta sentència enviem un objecte Java anomenat "pilota_gol". La mateixa funció pot ser utilitzada des del JESS per tal de dur a terme la mateixa acció d'enviar informació. La sintaxi en aquest cas és: (store <name> <value>)) Un exemple és: (store "fer" 1) I tal i com hem vist, estem enviant un objecte de nom "fer" que conté el valor 1.

Conjuntament amb aquesta funció existeix la funció *fetch* que realment podríem dir que és la funció inversa del *store*. Mitjançant aquesta funció el que fem és recollir un objecte que ens hagi estat enviat i obtenim el seu valor. La seva sintaxi en Java és: Public Value fetch(String name) I si la volem utilitzar des de JESS l'hem d'escriure com: (fetch <name>)

Com es pot observar el mecanisme de comunicació és realment senzill mitjançant l'ús d'aquestes dues funcions doncs són totalment complementàries. Si des de Java enviem un objecte mitjançant un *store*, el reco-

llim des de JESS amb la funció *fetch*, i de la mateixa forma si la comunicació es realitza en el sentit invers. Coneixent aquestes dues funcions veiem com seria la interacció habitual entre Java i JESS.

1.- Java envia les dades al JESS

Des de Java recollim la informació que requerim sobre el nostre entorn i el robot mateix, i la traspassem al JESS perquè aquest prengui una decisió.

Des de java enviem les dades:

```
JESS_engine.store("pilota_gol", new Value (pilo-
ta_gol.r, RU.FLOAT));
JESS_engine.store("pilota_gol_x", newValue
(LLARG_CAMP - Math.abs(pilota_gol.x),
RU.FLOAT));
```

Les quals recollim des de JESS:

```
(assert (pilota_gol (fetch "pilota_gol")))
(assert (pilota_gol_x (fetch "pilota_gol_x")))
```

2.- Java indica que comenci la inferència

Un cop hem enviat la informació que pugui haver de mester el SE, indiquem al mateix que s'executi. Podem aconseguir-ho utilitzant tres funcions:

- **Reset:** eliminem fets anteriors que puguin existir.
- **Assert:** inserim fets inicials.
- **Run:** executem el sistema expert.

3.- Java recull els resultats del JESS

Repetim de nou la comunicació store-fetch però en aquest cas en sentit contrari. És el JESS qui envia les dades, la decisió que cal prendre, cap al Java. Des de JESS faríem: `(store "fer" 0)`

En aquest cas la decisió presa està identificada amb el número 0. Obtenim aquest objecte des de Java utilitzant el fetch: `Value v = JESS_engine.fetch("fer");`

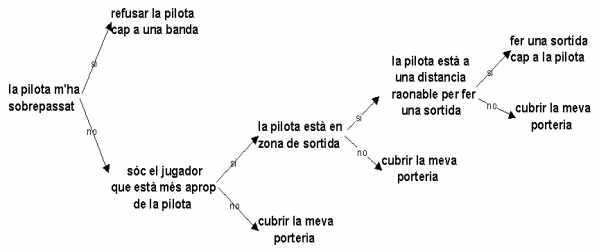
D'aquesta forma hem recollit la decisió en l'objecte *v*. Quan vulguem accedir al seu contingut ho podem fer mitjançant la funció *intValue*(). La qual ens retorna el valor associat a l'objecte, en el cas de l'exemple, 0.

4 Utilització del JESS per a la Decisió Intel·ligent d'un Jugador

Ara que ja sabem integrar IA al Javasoccer mitjançant el SE JESS, fem un exemple extret del curs: el porter *Higuita*. L'objectiu a l'hora de dissenyar la seva lògica de decisió ha estat donar-li un comportament força semblant al del veritable Higuita en la realitat. Aquest porter gairebé mai no és sota els pals per cobrir més espai de terreny i estar atent a possibles passades llargues dels rivals. Prefereix esperar des de la frontal de l'àrea. Des d'on aconsegueix reduir al màxim l'angle de tir de l'atacant rival, i pot sortir a refusar passades a l'espai buit de l'equip contrari. Pot semblar una política arriscada però s'ha seguit les pautes del mestre Cruyff en el Dream Team. D'altra banda, els jugadors no saben aixecar la pilota per sobre el porter.

4.1 Lògica de Decisió

El nostre porter pot prendre tres decisions diferents segons l'estat actual del joc. Aquestes són **a) Refusar a banda, b) Sortida i c)Cobrir porteria.** Anem a veure quan es decideix cadascuna d'elles amb les següents regles de decisió



4.2 Refusar a Banda

N'Higuita decideix refusar cap a banda quan la pilota l'ha sobrepassat que és quan la pilota es troba més aprop de la porteria que ell mateix. En aquesta situació considerem que cal que el porter corri cap a la pilota i intenti enviar-la cap al còrner abans no acabi dins la porteria. Podríem dir que és la darrera acció desesperada per intentar salvar el gol. I podem definir aquesta decisió mitjançant la següent regla en JESS:

```
(defrule refusar
        ?0<-(mes_aprop_pilota)
=>
        (assert (refusar))
        (retract ?0)
        (store "fer" 2) )
```

4.3 Sortida

El nostre porter més de jugar avançat li agrada sortir fora de l'àrea per jugar la pilota amb els peus i refusar aquesta cap al mig del camp. Tot i aquesta tendència no podem permetre que qui ens ha de salvar els gols es dediqui a sortir alegrement i tan sols li permetem que es prengui aquestes alegries quan considerem que la sortida és prou segura.

I què volem dir amb prou segura? Doncs que es compleixen les següents quatre condicions:

1.- La pilota es troba prou a prop de n'Higuita, en el que nosaltres hem anomenat distància de sortida. El porter tan sols ha de sortir cap a la pilota quan aquesta realment sigui *prou pròxima* a ell.

2.- La pilota es troba prou *a prop* de la porteria, dins el que podem anomenar la zona d'influència del porter. Mitjançant aquesta condició aconseguim limitar en certa manera la longitud de les sortides.

3.- N'Higuita és el jugador de l'equip *més proper* a la pilota. No podem esperar cap altre company hi vagi i de qualque manera hem de prendre la iniciativa.

4.- N'Higuita es troba també més a prop de la pilota que cap dels jugadors rivals.

Quan totes aquestes condicions són certes el nostre porter decideix sortir cap a la pilota. I podem definir aquesta decisió mitjançant la següent regla en JESS:

```
(defrule sortir
        (not (refusar))
        (distancia_sortida)
        ?0<-(pilota_dins_zona)
        ?1<-(visitant_mes_aprop)
        ?2<-(local_mes_aprop)
=>
        (assert (sortir))
        (retract ?0 ?1)
        (store "fer" 1)  )
```

4.4 Cobrir la Porteria

Finalment ens manca comentar la darrera possible alternativa que té el nostre porter, que consisteix en cobrir bé la porteria per reduir al màxim l'angle de tir dels rivals i evitar que cap xut el sorprengui.

Aquesta és de fet la decisió més habitual i es prendrà sempre que no es decideixi cap de les anteriors. És, per dir-ho d'una forma, la decisió comodí, doncs representa la tasca bàsica d'un bon porter.

Ens manca explicar un petit aspecte: com decideix el porter quina és la millor manera de cobrir la porteria? a quina posició s'ha de col·locar el porter? Llancem una línia imaginària des del centre de la nostra porteria cap a la pilota, en el que seria una previsió del xut més habitual del nostres rivals. El porter s'ha de col·locar sobre aquesta línia, i exactament en el punt que es troba a una distància de 0.32 respecte al centre de la porteria. Aquest valor s'ha trobat heurísticament.

En aquest cas la regla és ben senzilla de definir en el JESS. Indiquem que cal cobrir la porteria quan no es produeix cap de les altres decisions, és a dir: quan no sortim ni rebutgem. Veiem aquesta regla:

```
(defrule cubrir_porteria
        (not (refusar))
        (not (sortir))
=>
        (store "fer" 0) )
```

4.5 Implementació de la Lògica

Arribats aquest punt hem definit com es comporta el nostre porter Higuita, explicant quina decisió pren en funció del joc. Però tot ho hem fet parlant a un nivell molt alt i utilitzant expressions com "si és el jugador més a prop", "si la pilota és prou a prop seu"... fets que per nosaltres quan juguem a futbol són trivials de decidir si són certs o falsos, però que no és tan senzill quan els hem de dur a la implementació de la base de coneixement.

En aquest apartat el que volem és precisament explicar com calculem tots aquests fets que després ens permeten prendre la decisió més encertada en cada cas. Comencem per veure quines són les dades que inicialment ens transmet el programa Java cap el JESS.

4.6 Recollida d'Informació

Per poder començar aplicar les regles hem de comptar amb mesures sobre el nostre entorn. Ens referim a aspectes com el número de jugadors rivals, la distància que ens trobem de la pilota... Veiem concretament quina és aquesta informació imprescindible:

- **Pilota_porteria:** distància des de pilota cap el centre de la nostra porteria.
- **Pilota_porteria_f:** distància des de la posició futura de pilota cap el centre de la nostra porteria.
- **Jo_porteria:** distància que al centre porteria.
- **Nlocals:** número de components al nostre equip.
- **Nvisitants:** número de components a l'equip rival.
- **Dlocal0:** distància a la que em respecte la pilota.

Totes aquestes dades les obtenim des del nostre programa Java i les comuniquem al JESS mitjançant la funció *store*. Ja dins el JESS aquest pot recollir els valors utilitzant la funció *fetch*. Veiem com ho faria:

```
(defrule qui_pilota
        ?pos<-(fet obtenir_posicions)
=>
        (assert (pilota_porteria (fetch "pilota_porteria")))
        (assert (pilota_porteria_f (fetch "pilota_porteria_f")))
        (assert (jo_porteria (fetch "jo_porteria")))
        (assert (nlocals (fetch "nlocals")))
        (assert (nvisitants (fetch "nvisitants")))
        (assert (dlocal0 (fetch "dlocal0")))
        (retract ?pos) )
```

Ara bé, les dades fonamentals no són només aquestes. Ens en manquen de molt importants com poden ser la distància que es troben els altres jugadors de la pilota. Realment és informació que necessitem, però no ens n'hem pas oblidat sinó que la recollim d'una forma que poguem adaptar-nos tant al nombre de jugadors locals com de visitants. La resta de dades que ens manquen fan referència als nostres companys i als jugadors rivals. Imaginem-nos que recollim la distància que es troben de la pilota els jugadors de l'equip contrari dins aquesta mateixa regla que acabem de veure. Faríem:

```
(assert (dvisitant0 (fetch "dvisitant0")))
(assert (dvisitant1 (fetch "dvisitant1")))
    ... // i així amb la resta de dvisitanti
```

I obtindríem les dades referents als nostres cinc rivals. Però què passaria si l'equip rival decidís jugar amb tan sols quatre jugadors? El nostre programa petaria i perdríem el partit. Davant aquestes estratègies "estranyes" hem hagut d'optar per adaptar aquests "fetchs" al nombre de jugadors que en formen part.

La informació que interessa és:

- **Dlocal i:** distància del jugador ***i*** del nostre equip respecte la pilota.
- **Dvisitant i:** distància del jugador ***i*** de l'equip rival respecte la pilota.

```
(defrule entrada_local1
        (nlocals ?nl&:(> ?nl 1))
=>
        (assert (dlocal1 (fetch "dlocal1"))) )
(defrule entrada_local2
        (nlocals ?nl&:(> ?nl 2))
=>
        (assert (dlocal2 (fetch "dlocal2"))) )
... (i així fins tots els jugadors locals)
```

I l'aconseguim de la següent forma (mostrem tan sols el cas de la distància dels nostres companys, a l'hora de treballar amb els jugadors rivals actuaríem de la mateixa forma).

Fixem-nos que comprovem cada vegada quin és el nombre de jugadors locals, i segons aquest valor es dispararan més o menys regles que provoquin el *fetch* de la distància del jugador. També ens podem adonar com no considerem el cas que hi hagi un sol jugador local, i aquesta informació ja la recollim quan fem aquell primer conjunt de *fetchs* inicials que hem pogut veure. La raó és ben senzilla: com a mínim sempre hi ha un jugador local, nosaltres mateixos. Tot seguit anem a veure els aspectes de la situació del joc.

La Pilota Sobrepassa el Porter: mes_aprop_pilota

Quan la distància entre porteria i pilota és menor que la distància entre porter i la porteria, el porter ha estat sobrepassat i concloem un nou fet: *més_aprop_pilota*. Aquest fet vol dir que la pilota està més a prop de la porteria que el porter. Veiem la regla en JESS:

```
(defrule mes_aprop_pilota
        ?0<-(pilota_porteria ?pp)
        ?1<-(jo_porteria ?jp&:(<= ?pp ?jp))
=>
        (assert (mes_aprop_pilota))
        (retract ?0 ?1) )
```

La Pilota és a Zona d'Influència: pilota_dins_zona

Recordem com una de les condicions perquè el nostre porter decideixi sortir és que la pilota estigui dins el que hem anomenat la seva zona d'influència. El que realment volem dir és que la pilota està prou a prop de la nostra porteria. El codi de la regla és:

```
(defrule pilota_dins_zona
        ?0<-(pilota_porteria_f ?ppf&:(< ?ppf 0.5))
=>
        (assert (pilota_dins_zona))
        (retract ?0) )
```

Podem veure dos aspectes d'aquesta regla:

1.- Hem decidit que la pilota es troba dins la zona d'influència quan hi és a una distància inferior a 0.5.

2.- Treballem amb la distància futura de la pilota. És a dir, no ens estem fixant en on es troba la pilota ara mateix sinó que prenem la decisió tenint en compte a quina distància es trobarà la pilota d'aquí un temps.

D'aquesta forma aconseguim tenir en compte el sentit del moviment de la pilota. No és el mateix que la pilota estigui venint cap a la nostra porteria, que si aquesta s'està allunyant i fuig en direcció contrària.

La Sortida no és Excessiva: distancia_sortida

Un altre aspecte que cal considerar és evitar la sortida del nostre porter quan això suposa fer un recorregut massa llarg. El que fem és mirar a quina distància ens trobem respecte la pilota, si aquesta és inferior a 0.3

considerem que estem prou aprop de la mateixa, i inserim el nou fet: distancia_sortida.

```
(defrule distancia_sortida
        (dlocal0 ?dl0&:(< ?dl0 0.3))
=>
        (assert (distancia_sortida)) )
```

Sóc el Jugador Local més Aprop: local_mes_aprop

Un dels aspectes més importants a l'hora de prendre una decisió o altra és conèixer la situació dels nostres companys. En aquest sentit un dels aspectes més interessants és comprovar si sóc o no el jugador del meu equip que es troba més proper a la pilota.

La manera de conèixer si sóc o no el local més proper a la pilota és comparar la distància a la que em trobo de la pilota amb la distància a la que es troben els meus companys i veure si sóc la distància més petita.

Cal adaptar les nostres regles per tal que puguin funcionar sigui quin sigui el número de jugadors locals. El resultat és el següent:

```
(defrule local1_mes_aprop
        (nlocals ?nl&:(> ?nl 1))
        (dlocal0 ?dl0)
        ?a<-(dlocal1 ?dl1&:(> ?dl1 ?dl0))
=>
        (assert (local1_mes_aprop))
        (retract ?a)  )
(defrule no_local1_mes_aprop
        (nlocals ?nl&:(<= ?nl 1))
=>
        (assert (local_mes_aprop))  )
... i així successivament amb la resta de locals.
```

5 Conclusions del Curs

El curs es va impartir amb normalitat i els diversos grups vàrem comprovar de forma comparada les potencialitats de les diverses tècniques d'IA. Varem partir tots amb un coneixement zero del JavaSoccer, JESS i les altres tècniques d'IA i varen el 90% en quatre mesos preparar equips per a la competició i aprovar el curs. Les grans conclusions qu'ell n'extragueren del curs foren:

- Que la lògica difusa representava molt bé el coneixement imprecís i/o incert i simplificava molt el número de regles i situacions a generar.
- Que les xarxes neuronals són molt elegants per entrenar la decisió, i molt robustes si estan ben entrenades. Per contra es va experimentar el problema de la topologia (que encara és empírica) i la convergència.
- Els algorismes genètics són una alternativa raonable d'aprenentatge però conceptualment no han estat tan fàcils d'operar com les xarxes neuronals.
- Els algorismes on-line d'aprenentatge són eficaços però tendeixen a respondre (adaptar-se) molt tard en temps de resposta respecte a d'altres (l'equip qu'els va implementar va perdre tots els partits per golejada).
- El sistemes experts amb lògica difusa surten molt potenciats. Altrament són molt limitats.
- Aspectes de cooperació, les estratègies *deliberatives* són molt més eficients que les *reactives*. 2 equips per separat arriben a les mateixes observacions.

La conclusió general és que els alumnes han après molt bé i de forma *comparada* el que un mostreig de tècniques d'IA poden fer, i veuen els problemes d'incloure (embed) IA en sistemes per crear sistemes intel·ligents, que és un aspecte molt important per què en el futur puguin aplicar tècniques d'IA de forma rutinària dins de la seva carrera professional.

Evidentment que el que els ha motivat més ha estat la *competició*, anc que finalment crec que els he atrapat a la IA. Ara estan entusIAsmats per la IA i la majoria d'ells actualment fan projectes fi de carrera on apliquen qualques aspectes d'IA, el que dóna esperança que en el futur siguin receptius a continuar aplicant-la. Sobre tècniques concretes, les més populars entre els alumnes han estat les basades en la lògica difusa, possiblement perque faciliten enormement l'interfassatge numèrico-simbòlic necessari per a la decisió intel·ligent dels robots futbolistes.

Per acabar, animaria a altres professors que facin innovació dins d'aquesta línia doncs hi ha molt de treball futur, específicament en ampliar el repertori de tècniques d'IA aplicables, perfeccionar els mecanismes de comparació dels resultats dels diversos algorismes i poder observar aplicacions concretes d'algorismes primerament prototipats en el JavaSoccer i després aplicats a altres problemes anàlegs de la vida real.

Agraïments

Aquest treball ha estat finançat per la CICYT TAP98-0955-C03-02 dins del projecte "Diseño de agents físics (DAFNE)" així com la xarxa temàtica de la CIRIT tele-ensenyament i recerca en Automàtica- IITAP 1996 XT 00023. També agraïm que el nostre amic en

Tucker Balch de la CMU hagi donat freesoftware la darrera versió del seu *JavaSoccer*. Finalment vull agraïr als revisors anònims que mos han ajudat a millorar l'article docent.

6 Referències

[Asada 97] Asada M., Kuniyoshi Y., et al. The RoboCup Physical Agent Challenge, *In the First RoboCup Workshop in the XV IJCAI-97 International Joint Conference on Artificial Intelligence*, pp.51-56, 1997.

[Jennings 98] Jennings N., Sycara K., and Wooldridge M. A Roadmap of Agent Research and Development, *In Autonomous Agents and Multi-Agent Systems*, Kluwer Academic Publishers.Vol. 1, n. 1, pp. 7-38, 1998.

[Kitano 94] Kitano H. and Hondler J. A., Massively Parallel Artificial Intelligence, *In the AAAI Press / The MIT Press* pp: 1-52, *1994*

[Kumar xx] Kumar D. and Mawr B., Using Robots in an AI Course, Department of Math & Computer Science http://blackcat.brynmawr.edu/~dkumar/UGAI/Robots.html

[Mackworth 99] Mackworth A., The Dynamics of Intelligence: Constraint-Satisfying Hybrid Systems for Perceptual Agents, *In AAAI Spring Symposium in Hybrid Systems and AI*, 1999

[MacWorth 93] MacWorth A. K., Computer Vision: System, Theory, and Applications, *Chapter of On Seeing Robots, pp:1-13. World Scientific Press*, Singapore, 1993

[Mataric 98]Mataric M., Co-ordination and Learning in Multirobot Systems, *in the IEEE in Intelligent Systems*, March/April 1998, pp: 6-9

[Russell 94] Russell S. and Norvig P., *A Modern, Agent-Oriented Approach to AI Instruction. In Proceedings of the AAAI Fall Symposium on Innovative Instruction for Introductory AI*, New Orleans, Nov. 1994.

[Sahota 95] Sahota M. K., MacWorth A. K., Barman R. A., and Kingdon S. J., Real-Time Control of Soccer-Playing Robots using Off-Board Vision: the Dynamite Testbed, In *IEEE International Conference on Systems, Man and Cybernetics*, pp: 3690-3693, 1995.

[Steels 92, 97] Steels L., The Origins of Syntax in Visually Grounded Robotic Agents, *In the 15 IJCAI Proceedings*, pp: 1632-1641, 1997