**Inteligência artificial**

**Aplicação de C4.5 ao Diagnóstico de Doenças do Sistema Urinário**

Fábio Filipe Jesus Silva - ei11107

João Manuel Mesquita Cardoso – ei11100

Índice

Objetivo e especificação…………………………………………………3

Representação do conhecimento……………………………………….4

Trabalho efetuado, resultados esperados e conclusão……………….5

Recursos……………………………………………………………………6

Objetivo

O principal objetivo deste trabalho passa por usar algoritmos de aprendizagem (ID3 ou C4.5) para o diagnóstico de doenças (neste caso duas) do sistema urinário: infeção na bexiga ou nefrite de origem na pélvis renal. Tal é conseguido derivando regras de classificação com base em casos anteriores, determinando para isso a árvore de decisão que traduz tais regras.

Especificação

Como já fora dito, este trabalho passa por usar os algoritmos referidos para classificar doenças do sistema urinário, com base em atributos previamente definidos e passíveis de análise por parte dos mesmos algoritmos. De modo a atingir o objetivo, recorrer-se-á à utilização de uma biblioteca de Java que implementa o algoritmo C4.5, *WEKA* (*Waikato Environment for Knowledge Analysis*). Optou-se por este algoritmo pois permite a utilização de atributos tanto discretos como contínuos, entre outras melhorias em relação ao ID3.

Haverá uma separação em 4 módulos essenciais, cada qual com a sua tarefa específica.

Para a Representação do Conhecimento, teremos o módulo de leitura dos elementos do domínio em análise, que fará a leitura a partir de um ficheiro de texto. Tal ficheiro deve estar num formato específico, detalhado mais adiante.

O segundo módulo será responsável pela criação da árvore de decisão recorrendo à *WEKA*, utilizando para tal a classe *J48*, representativa da árvore de análise do algoritmo C4.5, para posterior treino.

O terceiro módulo abordará o treino e teste desta árvore. Para tal, dividiu-se o ficheiro original dos dados em dois novos ficheiros: *test.data*, com os dados de aprendizagem correspondentes a 2/3 do original (80 instâncias); e *train.data*, que possui dados para teste do algoritmo, correspondentes a 1/3 do original (40 instâncias). Esta abordagem (divisão de ficheiros) pareceu-nos a mais correta, uma vez que com esta divisão evita-se a necessidade de processamento extra para testes, quando há apenas um ficheiro indiferenciado.

O quarto e último módulo estará encarregue da identificação de novos casos com base nos sintomas que serão atempadamente pedidos ao utilizador via linha de comandos.

Em termos de ordem de desenvolvimento, abordar-se-á cada um dos módulos pela ordem descrita acima, pois achamos que será essa a logicamente correta para o decorrer do trabalho.

Haverá também a necessidade de pré-processamento de dados, uma vez que este possuem dois atributos classificativos. Logo, será necessário haver uma fusão de ambos ou a desconsideração de um deles.

Será também apresentado ao utilizador uma medição estatística dos resultados, assim como a representação da Árvore de Decisão resultante.

Representação do conhecimento

Os dados estarão albergados num ficheiro de texto, de modo a modularizar o programa e separar a informação da parte da resolução do problema.

O ficheiro organiza-se da seguinte maneira e pela sintaxe apresentada:

@relation disease //Aqui, o nome da relação é irrelevante

@attribute temperature numeric //Declaracao do atributo *temperature*, do tipo vírgula flutuante

@attribute nausea {yes, no} //Atributo náusea, do tipo nominal (sim ou não são as hipóteses)

@attribute lumbarPain {yes, no}

@attribute urinePushing {yes, no}

@attribute micturitionPains {yes, no}

@attribute burningOrItch {yes, no}

@attribute inflammationOfBladder {yes, no} //Classificador

@attribute nephritisOfRenalPelvis {yes, no} //Classificador

@data //Secção das instâncias

35,no,yes,no,no,no,no,no

35.9,no,no,yes,yes,yes,yes,no

//…

Como se pode ver por este excerto, o ficheiro contém duas partes fundamentais: *header*, onde se indica o tipo de dados que serão lidos (dados, atributos…) e a secção dos dados propriamente ditos.

Esta estrutura foi adotada porque facilita ao desenvolvimento do programa, uma vez que a *WEKA* exige que ficheiros lidos estejam neste formato.

Trabalho efetuado

Até ao momento da escrita deste relatório, a aplicação já se encontrava num estado consideravelmente avançado de desenvolvimento. Havia já leitura de dados, construção e treino da Árvore de Decisão, testes da mesma com uma eficácia de 100% e identificação correta de novos casos através da introdução de sintomas via *CLI* (*Command Line Interface*). No entanto, algumas alterações serão feitas no que toca aos atributos classificativos, uma vez que se apurou ser melhor a fusão ou desconsideração de um, o que não acontecia até ao momento (a árvore estava treinada tendo em conta um deles como um atributo regular).

Resultados esperados e forma de avaliação

No final do projeto, é de esperar que a aplicação corra o mais eficazmente possível. Com isto quer-se dizer que o programa acerte o maior número possível de casos, que otimamente será 100% destes. Para fazer tal avaliação, contamos com um ficheiro único de casos de teste para avaliação do algoritmo (resultado esperado *versus* resultado real).

Conclusão

No final deste projeto, esperamos que a nossa aplicação cumpra todos os requisitos definidos *a priori* mas, sobretudo, que possamos aprender os fundamentais acerca dos algoritmos ID3 e C4.5 (em especial este último), para que no futuro possamos aplicar corretamente esses conhecimentos em casos mais complexos e de diferentes dimensões, pois é sabido que têm uma vasta utilização no campo da Inteligência Artificial e são de extrema utilidade no mais diversos casos.

Recursos

* Biblioteca *WEKA v3.7* (http://weka.wikispaces.com/)
* Eclipse IDE
* Java 7
* Czerniak, Jacek , Ficheiro de dados para treino e teste (http://archive.ics.uci.edu/ml/datasets/Acute+Inflammations)