Ministério da
Ciência e Tecnologia

INPE-16640-RPQ/845

MINERAÇÃO DE DADOS APLICADO AO JOGO LIGA QUATRO

Wesley Gomes de Almeida

Relatório final da disciplina Princípios e Aplicações de Mineração de Dados (CAP-359) do Programa de Pós-Graduação em Computação Aplicada, ministrada pelo professor Rafael Santos.

Registro do documento original:
<http://urlib.net/sid.inpe.br/mtc-m18@80/2009/10.22.00.15>

INPE
São José dos Campos
2009

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3945-6911/6923
Fax: (012) 3945-6919
E-mail: pubtc@sid.inpe.br

SUMÁRIO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

1. INTRODUÇÃO

O jogo Liga Quatro também conhecido como *connect 4*, um jogo de tabuleiro para dois jogadores. Cada jogador coloca fichas em um tabuleiro começando de baixo para cima. As jogadas são feitas de forma alternada, ganha o jogador que conseguir conectar 4 fichas da mesma cor primeiro, em qualquer posição do tabuleiro: vertical, horizontal ou em uma das diagonais.

A Figura 1.1 ilustra um tabuleiro do jogo liga 4. Este tabuleiro é formado por seis posições na vertical e sete na horizontal. Neste exemplo, o jogador com fichas verdes venceu o jogo.

Este trabalho tem como principal objetivo estudar técnicas de mineração de dados para classificação de padrões de jogadas interessantes em uma partida do jogo liga 4.

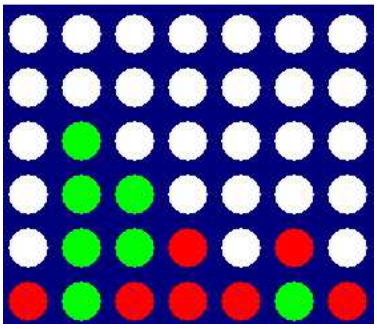

Figura 1.1– Tabuleiro do jogo liga 4

2. MATERIAL E MÉTODOS

O material utilizado para a elaboração desse projeto foi encontrado no site: *UCI Machine Learn Repository*. Estes dados correspondem aos oito primeiros movimentos de uma partida do jogo. O exemplar original possui 67557 instâncias e 42 atributos. Estes atributos correspondem a cada posição do tabuleiro do jogo.

A Figura 2.1 apresenta o formato do tabuleiro, as colunas são nomeadas pelas letras A, B, C, D, E e F, e as linhas por números inteiros variando de 1 a 6. Cada posição da

tabela é formada por uma letra maiúscula seguida por seu número de linha correspondente.

Com isso fica fácil o entendimento do formato dos dados de entrada. Pela Figura 2.2 verifica-se o formato de uma instância padrão, neste caso, tem-se 42 atributos, e uma variável *class*, esta indica se o jogo vai levar a uma vitória, derrota ou empate.

As Tabelas 2.1 e 2.2, indicam a ordem correta das posições do tabuleiro em uma linha do conjunto de dados. Neste caso, pela Tabela 2.2, o atributo 42 corresponde à célula G6 do tabuleiro apresentado na Figura 2.1 e o atributo 1 da tabela 2.1 representa o elemento A1.

6	A6	B6	C6	D6	E6	F6	G6
5	A5	B5	C5	D5	E5	F5	G5
4	A4	B4	C4	D4	E4	F4	G4
3	A3	B3	C3	D3	E3	F3	G3
2	A2	B2	C2	D2	E2	F2	G2
1	A1	B1	C1	D1	E1	F1	G1
	A	B	C	D	E	F	G

Figura 2.1– Posições do Tabuleiro

1, 2, 3, 4, 5, ..., 10, 11, ..., 20, 21, 22, ..., 30, 31, ..., 40, 41, 42, *class*

Figura 2.2– Sintaxe de uma instância

Tabela 2.1 – Dicionário de atributos de 1 a 24

1	A1	7	B1	13	C1	19	D1
2	A2	8	B2	14	C2	20	D2
3	A3	9	B3	15	C3	21	D3
4	A4	10	B4	16	C4	22	D4
5	A5	11	B5	17	C5	23	D5
6	A6	12	B6	18	C6	24	D6

Tabela 2.2 – Dicionário de atributos de 25 a 42

25	E1	31	F1	37	G1
26	E2	32	F2	38	G2
27	E3	33	F3	39	G3
28	E4	34	F4	40	G4
29	E5	35	F5	41	G5
30	E6	36	F6	42	G6

```
b,b,b,b,b,b,b,b,b,b,b,b,x,o,b,b,b,b,x,o,x,o,x,o,b,b,b,b,b,b,b,b,b,b,b,b,b,
b,b,b,b,b,win
b,b,b,b,b,b,b,b,b,b,b,b,x,b,b,b,b,b,x,o,x,o,x,o,o,b,b,b,b,b,b,b,b,b,b,b,b,
b,b,b,b,b,win
b,b,b,b,b,b,o,b,b,b,b,b,x,b,b,b,b,b,x,o,x,o,x,o,b,b,b,b,b,b,b,b,b,b,b,b,b,
b,b,b,b,b,win
b,b,b,b,b,b,b,b,b,b,b,b,x,b,b,b,b,b,x,o,x,o,x,o,b,b,b,b,b,b,o,b,b,b,b,b,b,
b,b,b,b,b,win
o,b,b,b,b,b,b,b,b,b,b,b,x,b,b,b,b,b,x,o,x,o,x,o,b,b,b,b,b,b,b,b,b,b,b,b,b,
b,b,b,b,b,win
b,b,b,b,b,b,b,b,b,b,b,b,x,b,b,b,b,b,x,o,x,o,x,o,b,b,b,b,b,b,b,b,b,b,b,b,o,
b,b,b,b,b,win
b,b,b,b,b,b,x,b,b,b,b,b,o,b,b,b,b,b,x,o,x,o,x,o,b,b,b,b,b,b,b,b,b,b,b,b,b,
b,b,b,b,b,draw
```

Figura 2.3– Exemplo de instâncias

Cada atributo pode conter um “x”, “o” ou “b”, sendo que “x” corresponde ao jogador 1, “o” ao jogador 2 e “b” significa que a posição do tabuleiro não foi preenchida. A Figura 2.3 corresponde as linhas iniciais do banco de informações utilizados para o desenvolvimento deste projeto.

Pelo exemplo da Figura 2.4 é possível verificar o resultado do mapeamento da instância apresentada na primeira linha do conjunto de testes da Figura 2.3.

Até o momento foi apresentando o formato dos dados originais disponíveis, porém neste projeto é proposto um outro formato de dados para facilitar a mineração e a identificação dos padrões de jogadas, possivelmente promissoras, para isso os dados originais devem sofrer um tipo de pré-processamento.

```
b,b,b,b,b,b,b,b,b,b,b,x,o,b,b,b,b,x,o,x,o,x,o,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,win
```

6			o				
5			x				
4			o				
3			x				
2		o	o				
1		x	x				
	A	B	C	D	E	F	G

Figura 2.4– Representação da instância 1

Na fase de pré-processamento é proposto uma janela 2×2, que tem por função fazer uma varredura nos dados originais a fim de identificar padrões e definir um novo formato de dados. Durante esta fase, a janela se responsabiliza em passar linha a linha procurando por padrões pré-definidos em uma tabela de dados. Ao encontrar um determinado padrão, esse deve ser identificado no novo conjunto de dados por seu índice correspondente definido na tabela de padrões.

A tabela de padrões utilizada corresponde a todas combinações possíveis de “b”, “x” e “o” em uma matriz 2×2, neste caso, existem 81 (3^4) padrões diferentes na tabela. Após a identificação dos padrões defini-se uma nova instância para os dados do jogo liga 4.

1	2	...	40	41	42	...	80	81	*class*
V	V	F	F	V	F	F	V	F	*draw*

Figura 2.5– Exemplo de instância

Como o objetivo desse trabalho é a identificação dos padrões que mais acontecem nas jogadas, a posição no tabuleiro onde eles ocorrem não importa. Por isso, a nova representação proposta pelo pré-processamento contém simplesmente os padrões

ocorridos, identificados por V (verdadeiro), quando o padrão é encontrado e F (falso), se o padrão não acontece.

Para o caso proposto neste projeto, o resultado do pré-processamento obtém instâncias com 81 atributos (cada um representando um padrão diferente) seguido por um atributo *class*. Cada um dos 81 atributos pode conter os símbolos V ou F, e o último, atributo *class* pode ter os valores: *win* (ganhou), *lost* (perdeu) ou *draw* (empatou). Pelo exemplo apresentado na Figura 2.5 verifica-se a ocorrência dos padrões 1, 2, 41 e 80, e o parâmetro *class=draw* indica que as jogadas podem levar a um empate.

Até o momento parece que a representação está correta, porém ao considerar que o pré-processamento tem como objetivo diminuir os dados por meio da utilização das informações mais importantes, verifica-se que não foi o que aconteceu. Com a representação original é possível enumerar 3^{42} testes diferentes, enquanto que pela representação proposta existem 2^{81} possibilidades. Com esses cálculos, surge uma dúvida, por que a filtragem aumentou a região viável?

Após a realização de diversos estudos, descobriu-se que o número de padrões igual a 81, está incluindo possibilidades inválidas para o jogo liga 4. Ao aplicar conceitos de matemática discreta é possível calcular o número correto de possibilidades para os padrões aceitos no jogo.

{x, o}	{x, o, b}	Caso 1		
b	{x, o}	2.3.1.2 = 12		
ou		+		
{x, o, b}	{x, o}	Caso 2		Total
{x, o}	b	3.2.2.1 = 12	=	32 casos inválidos
ou		+		
{x, o, b}	{x, o, b}	Caso 3		
b	b	3.3.1.1 = 9		

Figura 2.6 – Número de combinações inválidas

A regra principal diz que um “x” ou “o” nunca podem ocorrer em cima de um espaço em branco “b”. Logo, uma maneira fácil é pensar nos casos que não podem acontecer, e depois subtraí-los do número de casos total (3^4) . Na Figura 2.6 é apresentado os casos inválidos, com isso conclui-se que o número de possibilidades válidas correspondem a 3^4-32 = 49 padrões a serem analisados. Veja a enumeração dos casos na Tabela 2.3.

Baseado nessas análises o número de padrões utilizados é reduzido de 2^{81} para 2^{49}, com essa representação e com os padrões viáveis definidos é possível representar todos os padrões (2×2) previstos no jogo liga 4.

Para ajudar no processo de mineração, além da eliminação dos padrões inviáveis, eliminou-se o padrão um, porque este sempre acontece, uma vez que todos os testes sempre terão alguma janela 2×2 em branco. Com a filtragem dos dados e eliminação do padrão 1, obtém se um conjunto de dados menor. O exemplo apresentado na Figura 2.7 mostra as 6 primeiras instâncias após o pré-processamento.

```
F,F,V,F,V,F,V,F,V,F,F,F,F,F,F,V,F,F,F,F,F,F,F,F,F,F,F,V,F,F,F,F,F,F,F,F,F,
F,F,F,F,F,V,F,F,F,F,win
V,F,V,F,V,F,F,F,V,F,F,F,F,F,F,F,F,F,V,F,F,F,F,F,F,F,V,V,F,F,F,F,F,F,F,F,F,
F,F,F,F,V,F,F,F,F,F,win
F,F,V,F,V,F,V,F,V,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,V,F,V,F,F,F,F,F,F,F,
F,F,F,F,V,F,F,F,F,F,win
V,F,V,F,V,F,V,F,V,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,V,V,F,F,F,F,F,F,F,F,
F,F,F,F,V,F,F,F,F,F,win
V,F,V,F,V,F,F,F,V,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,V,V,F,F,F,F,F,F,F,F,
F,F,F,F,V,F,F,F,F,F,win
F,F,V,F,V,F,V,F,V,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,V,V,F,F,F,F,F,F,F,F,
F,F,F,F,V,F,F,F,F,F,win
```

Figura 2.7 – Exemplo de padrões

Tabela 2.3 – Tabela de padrões

1	2	3	4	5	6	7	8	9
b b b b	o b b b	x b b b	b o b b	o o b b	x o b b	b x b b	o x b b	x x b b
10	11	12	13	14	15	16	17	18
b b o b	o b o b	x b o b	b o o b	o o o b	x o o b	b x o b	o x o b	x x o b
19	20	21	22	23	24	25	26	27
b b x b	o b x b	x b x b	b o x b	o o x b	x o x b	b x x b	o x x b	x x x b
28	29	30	31	32	33	34	35	36
b b b o	o b b o	x b b o	b o b o	o o b o	x o b o	b x b o	o x b o	x x b o
37	38	39	40	41	42	43	44	45
b b o o	o b o o	x b o o	b o o o	o o o o	x o o o	b x o o	o x o o	x x o o
46	47	48	49	50	51	52	53	54
b b x o	o b x o	x b x o	b o x o	o o x o	x o x o	b x x o	o x x o	x x x o
55	56	57	58	59	60	61	62	63
b b b x	o b b x	x b b x	b o b x	o o b x	x o b x	b x b x	o x b x	x x b x
64	65	66	67	68	69	70	71	72
b b o x	o b o x	x b o x	b o o x	o o o x	x o o x	b x o x	o x o x	x x o x
73	74	75	76	77	78	79	80	81
b b x x	o b x x	x b x x	b o x x	o o x x	x o x x	b x x x	o x x x	x x x x

Padrões Inviáveis Padrões Possíveis

3. RESULTADOS

Os dados gerados por esse projeto foram convertidos para o formato “.arff”, extensão conhecida pelo software Weka. Tal software contém um conjunto de algoritmos de aprendizado para tarefas de mineração de dados. As tarefas realizadas por esses algoritmos incluem pré-processamento, classificação, regressão, agrupamento, regras de associação e visualização.

Os testes executaram em um computador com processador Core2-Duo 2.0 GHz, com 2 GBs de memória RAM, sob o sistema operacional Linux.

Para os testes utilizou-se 2 algoritmos, o J48 método de classificação em árvores e o Apriori, método de busca por regras de associação. A Figura 3.1 apresenta os resultados após a execução do J48, como pode ser visto pela matriz de confusão o resultado não obteve sucesso, e o número de níveis da árvore foi muito grande.

Na tentativa de encontrar regras de associação executou-se o algoritmo Apriori. A Figura 3.2 mostra as principais regras obtidas pelo método.

A Figura 3.3 apresenta as regras de associação obtidas, essas regras possuem um grau de confiabilidade fixado entre 0,92 e 0,73. De acordo com os resultados apresentados verifica-se que no caso da primeira regra, sempre que o padrões 10 e 58 ocorrem, o padrão 28 também acontece com uma confiabilidade de 0.92. Outro regra interessante diz com grau de confiança igual a 0,73 que quando o padrões 19 e 55 acontecem, a partida pode terminar com uma vitória de “x”.

```
Number of Leaves  :  1621

Size of the tree :   3241


Time taken to build model: 16.64 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       49905               73.871  %
Incorrectly Classified Instances     17652               26.129  %
Kappa statistic                          0.4081
Mean absolute error                      0.2426
Root mean squared error                  0.3631
Relative absolute error                 73.2314 %
Root relative squared error             89.2305 %
Total Number of Instances            67557

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.905     0.493      0.78      0.905     0.838      0.79     win
                 0.559     0.106      0.632     0.559     0.593      0.809    loss
                 0.054     0.014      0.287     0.054     0.091      0.645    draw
Weighted Avg.    0.739     0.352      0.696     0.739     0.706      0.781

=== Confusion Matrix ===

     a     b     c   <-- classified as
 40265  3768   440 |     a = win
  6917  9291   427 |     b = loss
  4467  1633   349 |     c = draw
```

Figura 3.1 – Resultados do algoritmo J48

```
  1. P39:=F P81:=F 61831 ==> P41:=F 61606    conf:(1)
  2. P43:=F P81:=F 61635 ==> P41:=F 61410    conf:(1)
  3. P39:=F 62246 ==> P41:=F 62011    conf:(1)
  4. P43:=F 62050 ==> P41:=F 61815    conf:(1)
  5. P51:=F P71:=F P81:=F 62001 ==> P41:=F 61731    conf:(1)
  6. P65:=F 61240 ==> P41:=F 60972    conf:(1)
...
998. P68:=F P69:=F P81:=F 61383 ==> P41:=F 61025    conf:(0.99)
999. P71:=F P78:=F P80:=F 61376 ==> P81:=F 61018    conf:(0.99)
1000. P42:=F P45:=F P71:=F 61375 ==> P81:=F 61017    conf:(0.99)
```

Figura 3.2 – Resultados do Apriori para confiabilidade entre 1 e 0.99

```
1. P10:=V P58:=V 33006 ==> P28:=V 30437    conf:(0.92) lift:(1.15) lev:(0.06)
[3892] < conv:(2.51)>
  2. P46:=F P58:=V 33394 ==> P28:=V 30509    conf:(0.91) lift:(1.14)
lev:(0.05) [3652] < conv:(2.27)>
  3. P20:=V P28:=V 35567 ==> P10:=V 32531    conf:(0.91) lift:(1.13)
lev:(0.06) [3817] < conv:(2.26)>
  4. P47:=F P58:=V 33714 ==> P28:=V 30696    conf:(0.91) lift:(1.13)
lev:(0.05) [3582] < conv:(2.19)>
  5. P19:=V P28:=V P66:=F 35320 ==> P46:=F 32116    conf:(0.91) lift:(1.12)
lev:(0.05) [3342] < conv:(2.04)>
…
13. P55:=V Class:=win 34962 ==> P19:=V 30926    conf:(0.88) lift:(1.13)
lev:(0.05) [3591] < conv:(1.89)>
…
44. P46:=F Class:=win 36222 ==> P19:=V 31515    conf:(0.87) lift:(1.11)
lev:(0.05) [3195] < conv:(1.68)>
…
104. P28:=V Class:=win 34383 ==> P47:=F 31753    conf:(0.92) lift:(1.05)
lev:(0.02) [1388] < conv:(1.53)>
…
144. P19:=V Class:=win 37315 ==> P55:=V 30926    conf:(0.83) lift:(1.1)
lev:(0.04) [2937] < conv:(1.46)>
…
193. P37:=F P67:=F Class:=win 36109 ==> P47:=F 33116    conf:(0.92)
lift:(1.04) lev:(0.02) [1227] < conv:(1.41)>
…
326. Class:=win 44473 ==> P47:=F 40537    conf:(0.91) lift:(1.03) lev:(0.02)
[1261] < conv:(1.32)>
…
447.  P19:=V  P55:=V  42266  ==>  Class:=win  30926      conf:(0.73)  lift:(1.11)
lev:(0.05) [3102] < conv:(1.27)>
…
450. P19:=V P47:=F P67:=F 42169 ==> Class:=win 30851    conf:(0.73)
lift:(1.11) lev:(0.05) [3091] < conv:(1.27)>
```

Figura 3.3 – Resultados do Apriori para confiabilidade entre 0.92 e 0.73

4. CONCLUSÕES

O idéia proposta neste trabalho é de grande relevância para diversas áreas, entre elas teoria de jogos, além de poder ser estendida para diversas áreas da computação.

Pelos testes apresentados é possível verificar que o método foi capaz de buscar por regras na base de dados proposta, em alguns casos obteve resultados até interessantes, entre eles regras que mostram padrões de jogadas que possam levar a vitória.

Com os resultados obtidos é possível desenvolver um sistema inteligente que implemente todas as regras importantes para o jogo liga quatro, em aplicações de jogos pode ser de grande relevância o uso desses tipos de regras, tais regras podem favorecer o agente envolvido.

Durante o desenvolvimento deste trabalho vários testes foram realizados. As tentativas iniciais incluíam execuções com todos os 81 padrões possíveis, porém com esse volume de informações a execução do método tornou-se inviável devido a limitações de memória e processamento.

BIBLIOGRAFIA CONSULTADA

http://archive.ics.uci.edu/ml/datasets/Connect-4
http://www.cs.waikato.ac.nz/ml/weka/