Departamento de Engenharia Informática e de Sistemas

Instituto Superior de Engenharia de Coimbra

Licenciatura em Engenharia Informática

- Conhecimento e Raciocínio -

2018/2019

Trabalho Prático

Trabalho realizado por:

Gabriel Pinheiro nº21260736

Guilherme Silva nº 2126031

CBR

O CBR (*Case Based Reasoning*) é um modelo utilizado em diversos sistemas periciais. Tem como base a utilização de diversos casos/experiências passadas em que cada um destes casos é composto por uma descrição e a respetiva solução. A resolução de novos casos é feita ao encontrar um ou mais casos semelhantes ao atual e apresentar uma solução adaptada ao novo contexto.

O Nosso Problema

O problema apresentado neste trabalho é o de perceber se uma pessoa tem tendência futura para contrair diabetes. Foi-nos fornecido um *dataset* com 768 exemplos de indivíduos com e sem diabetes. Os atributos têm o seguinte significado:

- *PregnanciesNumber of times pregnant*
- *GlucosePlasma glucose concentration a 2 hours in an oral glucose tolerance test*
- *BloodPressureDiastolic blood pressure (mm Hg)*
- *SkinThicknessTriceps skin fold thickness (mm)*
- *Insulin2-Hour serum insulin (mu U/ml)*
- *BMIBody mass index (weight in kg/(height in m)^2)*
- *DiabetesPedigreeFunction: Diabetes pedigree function*
- *AgeAge (years)*
- *OutcomeClass variable (0 or 1) 268 of 768 are 1, the others are 0*

CBR Aplicado Ao Nosso Problema

Inicialmente, após ler e guardar todos os casos do dataset disponibilizado, entramos na fase de *retrieve*, onde tivemos que atribuir *weighting factors* para os diferentes atributos, sendo que, ao pedir a opinião de pessoas que estão ativamente envolvidas no campo da medicina e enfermagem, chegamos à conclusão que iríamos usar uma escala de 1 a 5 do menos relevante ao mais relevante respetivamente. Estes são os valores que foram utilizados em todo o decorrer da experiência:

- *Pregnancies – 1*
- *Glucose - 3*
- *BloodPressure - 2*
- *SkinThicknessTriceps – 1*
- *Insulin - 3*
- *BMI - 5*
- *DiabetesPedigreeFunction - 1*
- *Age – 3*

Após esta atribuição ser feita já se pode comparar o novo caso com os casos já existentes e perceber quais as suas similaridades ou disparidades. Este processo é feito a partir do cálculo das distâncias absolutas entre os diferentes atributos e fazendo o cálculo da semelhança final entre os casos analisados, como podemos ver nos seguintes pedaços de código:

```
%É repetido o mesmo para todos os atributos
distances(1,1) = calculate_absolute_distance(…
                case_library{i,'Pregnancies'} / max_values('Pregnancies'), …
                new_case.Pregnancies / max_values('Pregnancies'));


final_similarity = 1-((distances*weighting_factors)/sum(weighting_factors));

        if final_similarity >= similarity_threshold
            retrieved_indexes = [retrieved_indexes i];
            similarities = [similarities final_similarity];
        end
```

Como podemos ver, todas as similaridades e todos os índices dos casos semelhantes são guardados se a semelhança for maior que o *similarity_threshold* que neste caso será de 0.75.

Depois disto segue-se para a fase de *reuse*, que consiste em reutilizar a informação dos casos obtidos de forma a resolver o problema. Para isso vamos utilizar os atributos dos casos semelhantes e através desta fórmula calcular o *Outcome*:

```
x1 = retrieved_cases{:,1}; % Pregnancies
x2 = retrieved_cases{:,2}; % Glucose
x3 = retrieved_cases{:,3}; % BloodPressure
x4 = retrieved_cases{:,4}; % SkinThickness
x5 = retrieved_cases{:,5}; % Insulin
x6 = retrieved_cases{:,6}; % BMI
x7 = retrieved_cases{:,7}; % DiabetesPedigreeFunction
x8 = retrieved_cases{:,8}; % Age

y = retrieved_cases{:,9};   %Outcome

X = [ones(size(x1)) x1 x2 x3 x4 x5 x6 x7 x8];

b = X\y;

outcome = b(1) + b(2) * new_case.Pregnancies + b(3) * new_case.Glucose + …
b(4) * new_case.BloodPressure + b(5) * new_case.SkinThickness + …
b(6) * new_case.Insulin + b(7) * new_case.BMI + …
b(8) * new_case.DiabetesPedigreeFunction + b(9)*new_case.Age;
```

Depois destas contas serem feitas vamos obter um valor entre 0 e 1 e a partir desse momento já temos o nosso *Outcome* (se for maior que 0.5 vai ser 1 e 0 caso contrário)

No *revise* apenas damos a escolher ao utilizador se ele deseja guardar o *Outcome* obtido ou não, dado que já sabemos que o valor vai ser válido antes de entrar nesta função, logo não é necessário fazer essa verificação.

APLICAÇÃO DE LÓGICA *FUZZY* AO NOSSO PROBLEMA

Para além de utilizar um modelo de CBR isolado, também recorremos à Lógica *Fuzzy* para ter uma maior aproximação de valores e um sistema mais coerente com aquilo que é a realidade. No nosso caso, definiram-se variáveis linguísticas para cada atributo (pregnancies, glucose, insulin, etc.).

```
    pedigree_low = [];          glucose_low = [];        Outcome = [];
    pedigree_med = [];          glucose_med = [];
    pedigree_high = [];         glucose_high = [];

    pregnan_low = [];
    pregnan_med = [];           thick_low = [];
    pregnan_high = [];          thick_high = [];

    press_low = [];             Age_young = [];
    press_med = [];             Age_youngadult = [];
    press_high = [];            Age_adult = [];
    press_crit = [];            Age_elder = [];

    insulin_low = [];           BMI_low = [];
    insulin_med = [];           BMI_med = [];
    insulin_high = [];          BMI_high = [];
```

Após esta definição, os valores da biblioteca são passados por funções gaussianas com os seguintes parâmetros para cada atributo (recorreu-se ao *Fuzzy Toolbox* para ter uma melhor noção do comportamento dos valores em cada função de pertença):

```
if case_library{i,'SkinThickness'} <= 0
            fuzified_thick = [0 0];
        else
            fuzified_thick = [gaussmf(case_library{i,'SkinThickness'},[10
17.5]) ...
                              gaussmf(case_library{i,'SkinThickness'},[10
50])];
        end
if case_library{i,'DiabetesPedigreeFunction'} <= 0
            fuzified_pedigree = [0 0 0];
else
            fuzified_pedigree =
[gaussmf(case_library{i,'DiabetesPedigreeFunction'},[0.2 0.275]) ...

gaussmf(case_library{i,'DiabetesPedigreeFunction'},[0.2 1]) ...

gaussmf(case_library{i,'DiabetesPedigreeFunction'},[0.2 2])];
        end


fuzified_preg = [gaussmf(case_library{i,'Pregnancies'},[1.5 2.5]) ...
                 gaussmf(case_library{i,'Pregnancies'},[1.5 7.5]) ...
                 gaussmf(case_library{i,'Pregnancies'},[1.5 15])];
```

Após passar pela função de pertença, cada atributo fica dividido em N parâmetros, sendo N o número de variáveis linguísticas definidas por atributo.

Face à fuzificação dos valores existentes, agora é necessário fazer o mesmo para o novo caso a ser testado.
Aplicam-se as mesmas variáveis linguísticas para cada atributo do novo caso, juntamente com a passagem desses valores pelas mesmas funções gaussianas que a biblioteca de testes teve efeito.
Dado estes valore, o sistema CBR funciona da mesma forma, porém com mais atributos que anteriormente. Assim sendo, existe mais termos de comparação entre a biblioteca de casos e o novo caso, calculando as distâncias/semelhanças entre cada atributo.

TESTES E CONCLUSÕES

Preg	Gluc	BPressure	SThickness	Insulin	BMI	DPF	Age	Outcome	CBR + *Fuzzy* Logic	CBR
0	78	88	29	40	36.9	0.434	21	0	-0.05	0.014
0	113	76	0	0	33.3	0.278	23	1	0.186	0.124
0	117	66	31	188	30.8	0.493	22	0	0.252	0.209
0	180	78	63	14	59.4	2.42	25	1	0.81	1.156
0	189	104	25	0	34.3	0.435	41	1	0.882	0.615
1	97	64	19	82	18.2	0.299	21	0	0.044	-0.09
1	112	80	45	132	34.8	0.217	24	0	0.162	0.185
1	122	90	51	220	49.7	0.325	31	1	0.390	0.429
1	139	62	41	480	40.7	0.536	21	0	0.477	0.428
1	155	84	44	545	38.7	0.619	34	0	0.75	0.574
2	56	56	28	45	24.2	0.332	22	0	-0.225	-0.211
2	102	86	36	120	45.5	0.127	23	1	0.05	0.244
2	125	60	20	140	33.8	0.088	31	0	0.39	0.313
2	142	82	18	64	24.7	0.761	21	0	0.220	0.299
2	155	52	27	540	38.7	0.24	25	1	0.276	0.499
2	175	88	0	0	22.9	0.326	22	0	0.534	0.410
3	78	50	32	88	31	0.248	26	1	-0.5	0.040
3	80	82	31	70	34.2	1.292	27	1	0.092	0.216
3	107	62	13	48	22.9	0.678	23	1	0.003	0.144
3	163	70	18	105	31.6	0.268	28	1	0.558	0.508
3	171	72	33	135	33.3	0.199	24	1	0.566	0.522
4	76	62	0	0	34	0.391	25	0	-0.004	0.090
4	103	60	33	192	24	0.966	33	0	0.308	0.219
4	183	0	0	0	28.4	0.212	36	1	0.983	0.787
4	197	70	39	744	36.7	2.329	31	0	0.743	1.136
5	189	60	23	846	30.1	0.398	59	1	0.647	1.135
7	81	78	40	48	46.7	0.261	42	0	0.107	0.338

7	83	78	26	71	29.3	0.767	36	0	0.283	0.180
7	136	74	26	135	26	0.647	51	0	0.571	0.490
7	148	60	27	318	30.9	0.15	29	1	0.509	0.432
7	160	54	32	175	30.5	0.588	39	1	0.835	0.682
7	195	70	33	145	25.1	0.163	55	1	0.87	0.737
8	186	90	35	225	34.5	0.423	37	1	0.991	0.791
9	123	100	35	240	57.3	0.88	22	0	0.466	0.624
9	134	74	33	60	25.9	0.46	81	0	-0.813	0.559
9	154	78	30	100	30.9	0.164	45	0	0.621	0.558
10	101	76	48	180	32.9	0.171	63	0	0.167	0.366
10	148	84	48	237	37.6	1.001	51	1	0.728	0.772
12	92	62	7	258	27.6	0.926	44	1	0.26	0.374
12	165	76	43	255	47.9	0.259	26	0	0.622	0.806
TOTAL Correct:		**CBR:** 27	**CBR + *Fuzzy*:** 25							

Dado os 40 testes efetuados, chegou-se à conclusão existe uma taxa de correção de por volta de 67.5% para o algoritmo de CBR *standalone*, e 62,5% para o modelo CBR com lógica *Fuzzy*.

Com isto, podemos concluir que por haver uma fuzificação dos valores, existem mais termos de comparação entre os valores da biblioteca de casos e os casos de teste. Isto poderia dar uma ideia de que com mais valores existiria uma maior eficácia em termos de previsão, porém, neste caso isso não se verificou.