Exercício de Fixação de Conceitos 3

EA072 - Inteligência Artificial em Aplicações Industriais

Caio **CRUVINEL FINARDI - RA 116342**
Gustavo **CIOTTO PINTON - RA 117136**

Campinas, 30 de novembro de 2015

1. Curvas *Precision-Recall* e ROC

De acordo com o *paper* de Davis & Goadrich (2006), os conceitos são definidos conforme a seguir:

a. ***Matriz de Confusão****:* as decisões tomadas por um classificador podem ser representadas nesta estrutura. Segundo o *paper*, para um problema de decisão binário, tal matriz possui quatro diferentes categorias, sendo elas *verdadeiros positivos (TP), falsos positivos (FP), negativos verdadeiros (TN)* e *falsos negativos (FN).* A primeira se refere aos exemplos preditos como verdadeiros e que efetivamente o são e a segunda categoria contém os exemplos preditos como verdadeiros mas que, na realidade são falsos. Por fim, a categoria *TN* diz respeito aos exemplos atribuídos corretamente como falso pelo modelo e a *FN* contém os exemplos ditos como negativo, mas que realmente deveriam ser avaliados como positivos.

b. ***Recall:*** esse medidor calcula a fração dos exemplos positivos que foram avaliados corretamente como positivos. Tem-se portanto:

$$RECALL = \frac{TP}{Total\ Positives} = \frac{TP}{TP+FN} \quad (1)$$

em que TP é o total de exemplos avaliados como positivos pelo modelo e *Total Positives* é a quantidade de exemplos que são efetivamente positivos.

c. ***Precision:*** esse medidor mede, por sua vez, a fração dos exemplos classificados como positivos e que realmente o são. A equação é representada logo abaixo:

$$PRECISION = \frac{TP}{FP+TP} \quad (2)$$

em que FP é a quantidade de exemplos avaliados erroneamente como positivos pelo modelo.

d. ***Taxa de verdadeiros positivos***: equivalente ao conceito de *Recall.* Conforme explicado anteriormente, esse medidor mede a taxa de acerto de exemplos considerados positivos pelo modelo.

e. ***Taxa de falsos positivos (FPR):*** esse medidor mede a fração de exemplos negativos que foram classificados incorretamente como positivos pelo sistema. É dado pela equação abaixo:

$$FPR = \frac{FP}{Total\ Negatives} = \frac{FP}{FP+TN} \quad (3)$$

em que TN é a quantidade de exemplos negativos avaliados corretamente.

f. ***Curvas Precision-Recal (PR)***: a curva *Precision-Recall* relaciona estes dois medidores, de forma que o *eixo y* mostra valores de *Precision* e o *eixo x*, de *Recall*. O objetivo neste tipo de gráfico é que as curvas se aproximem ao máximo ao canto superior *direito*, isto é, ambos os indicadores próximos de 1. Isto garante, pela equação (1), que o número de falsos negativos seja pequeno e, pela (2), que a quantia de falsos positivos também não seja considerável.

g. ***Curvas ROC:*** a curva ROC, do inglês *"Receiver Operator Characteristic"*, relaciona os medidores *FPR,* no eixo x, e TPR, no eixo y. O objetivo a ser atingido é que as curvas se aproximem maximamente ao canto superior *esquerdo,* isto é, FPR próximo de zero e TPR, de 1. Estes dois fatos garantem, segundo equações (3) e (1), respectivamente, que os números de falsos positivos e falsos negativos são muito pequenos. Ainda segundo o *paper*, tais curvas podem apresentar uma visão otimista, às vezes errônea, da performance de um algoritmo. Os autores demonstram

igualmente que uma curva domina no espaço ROC se e somente se esta curva domina no espaço *Precision-Recall,* e que um algoritmo que otimiza a área abaixo da curva ROC não otimiza garantidamente a área abaixo de uma curva PR.

h. ***Critério de desempenho AUC-ROC***: a área abaixo das curvas podem ser utilizadas como medidas da performance de um algoritmo. A área abaixo de uma curva ROC, ou *area under ROC curve (AUC-ROC)*, pode ser calculada pelo método dos trapézios, de forma a calcular a soma das áreas do trapézios formados entre dois pontos da curva.

Em um problema de classificação binária, pode não ser suficiente monitorar apenas a taxa de acerto do classificador pelo motivo de que o número de exemplos negativos da aplicação pode superar fortemente o número de exemplos positivos. Logo, uma grande mudança no número de falsos positivos pode levar uma pequena variação na taxa de falsos positivos (*FPR*) numa curva ROC. O medidor *Precision* elimina este problema, já que é capaz de capturar o efeito de um eventual grande número de exemplos negativos na performance do algoritmo, à medida que compara a quantidade de falsos positivos àquela de verdadeiros positivos ao invés de comparar ao número de verdadeiros negativos.

2. Árvores de Decisão

Os atributos encontrados são 7: temperatura média, umidade média, altura chuva mensal, precipitação 21 dias, numero dias de chuva, local e classe. Existem atributos númericos como Temperatura méda e atributos categóricos como local. Após executar o programa com as configurações sugeridas no enunciado, obtem-se:

64 classificados corretamente (91.4286%) e 6 classificados incorretamente (8.5714%).

```
=== Confusion Matrix ===
  a  b
  8  3  |  a = 0
  3 56  |  b = 1
```

A árvore final contem 18 folhas e 26 nós no total.

A matrix de confusão é utilizada para descrever performance de clusters e modelos de calssificação, ela é dividida da seguinte forma:

Tabela 1: matriz de confusão.

Positivos verdadeiros	Falsos negativos
Falsos positivos	Negativos verdadeiros

Portanto podemos notar que 64(8+56) foram classificados corretamente, 3 foram falsos positivos e 3 falsos negativos.

A árvore gerada pode ser descrita da seguinte forma:

```
SE <temperatura_media <= 21.87  e numero_dias_chuva <=4  e local=ID>  ENTAO < 1>
SE <temperatura_media <= 21.87  e numero_dias_chuva <=4  e local=IM>  ENTAO < 1>
SE <temperatura_media <= 21.87  e numero_dias_chuva <=4 e local=PM e numero_dias_chuva <=3.0> ENTAO
<0>
```

```
SE <temperatura_media <= 21.87  e numero_dias_chuva <=4 e local=PM e numero_dias_chuva >3.0> ENTAO
<0>
SE <temperatura_media <= 21.87  e numero_dias_chuva <=4 e local=GAL>  ENTAO < 0>
SE <temperatura_media <= 21.87  e numero_dias_chuva >4  e local=ID>  ENTAO < 1>
SE <temperatura_media <= 21.87  e numero_dias_chuva >4  e local=IM e umidade_media <= 97.5 >  ENTAO
<1>
SE <temperatura_media <= 21.87  e numero_dias_chuva >4  e local=IM e umidade_media > 97.5 >  ENTAO <
0>
SE <temperatura_media <= 21.87  e numero_dias_chuva >4  e local=PM>  ENTAO < 1>
SE <temperatura_media <= 21.87  e numero_dias_chuva >4  e local=CHI>  ENTAO < 1>
SE <temperatura_media <= 21.87  e numero_dias_chuva >4  e local=GAL>  ENTAO < 1>
SE <temperatura_media > 21.87  e local=ID>  ENTAO < 1>
SE <temperatura_media > 21.87  e local != ID>  ENTAO < 0>
```

O atributo Unpruned gerará arvores maiores, pois não passará por um algorítimo de poda. Além da vantagem de que árvores podadas são menores e mais compreensíveis, também existe a vantagem de evitar possíves overfitting, entretanto é possível que o erro aumente um pouco.

Ao se mudar o Unpruned para FALSE, uma árvore podada é gerada. Esta é claramente bem menor e muito mais simples, contendo apenas 2 folhas e uma raiz. O erro aumenta um pouco,12.8571%, aproximadamente 4% maior que a árvore sem poda.

O parametro minNumObj é o minimo de dados separados a cada branching. Ao se aumentar minNumObj para 5 , também resulta em uma arvore bem menor, com apenas 2 folhas(mesmo sem podar) e um erro também maior (15,71%), aproximadamente 7% a mais de erro.

Utilizando o arquivo vote.arff (Disponível em http://storm.cis.fordham.edu/~gweiss/data-mining/datasets.html) que tenta classificar politicos em Republicanos ou democratas com base em votos passados de cada político em 17 diferentes categorías listadas abaixo:

```
1. Class Name: 2 (democrat, republican)
%    2. handicapped-infants: 2 (y,n)
%    3. water-project-cost-sharing: 2 (y,n)
%    4. adoption-of-the-budget-resolution: 2 (y,n)
%    5. physician-fee-freeze: 2 (y,n)
%    6. el-salvador-aid: 2 (y,n)
%    7. religious-groups-in-schools: 2 (y,n)
%    8. anti-satellite-test-ban: 2 (y,n)
%    9. aid-to-nicaraguan-contras: 2 (y,n)
%   10. mx-missile: 2 (y,n)
%   11. immigration: 2 (y,n)
%   12. synfuels-corporation-cutback: 2 (y,n)
%   13. education-spending: 2 (y,n)
%   14. superfund-right-to-sue: 2 (y,n)
%   15. crime: 2 (y,n)
%   16. duty-free-exports: 2 (y,n)
%   17. export-administration-act-south-africa: 2 (y,n)
```

Nota-se que todos atributos são categóricos podem ser apenas Sim ou Não.Utilizando uma configuração para o J48 similar ao sugerido pelo roteiro, apenas com a mudação de unpruned para false, o que permite melhor compreensão da árvore, gerou-se a seguinte árvore de decição:

```
      physician-fee-freeze = n: democrat (253.41/3.75)
      physician-fee-freeze = y
      |   synfuels-corporation-cutback = n: republican (145.71/4.0)
      |   synfuels-corporation-cutback = y
      |   |   mx-missile = n
      |   |   |   adoption-of-the-budget-resolution = n: republican (22.61/3.32)
      |   |   |   adoption-of-the-budget-resolution = y
      |   |   |   |   anti-satellite-test-ban = n: democrat (5.04/0.02)
      |   |   |   |   anti-satellite-test-ban = y: republican (2.21)
      |   |   mx-missile = y: democrat (6.03/1.03)

      Number of Leaves  :     6

      Size of the tree :      11
```

Ao análisar a árvore é possivel visualizar os tópicos mais importantes na decisão entre Republicano e democrata, sendo eles: physician-fee-freeze, synfuels-corporation-cutback, mx-missile, adoption-of-the-budget-resolution,anti-satellite-test-ban.

Foram corretamente classificados 96.32% dos dados, com a seguinte matriz de confusão:

```
=== Confusion Matrix ===

   a   b   <-- classified as
 259   8 |   a = democrat
   8 160 |   b = republican
```

3. Redes Bayesianas

Executando o programa *bayes* em MATLAB, os seguintes resultados são encontrados:

1. Para analisar os dados presentes no arquivo `dados_BN_EFC3.mat`, escrevemos o *script* abaixo:

Programa 1: Analisa os dados a serem utilizados posteriormente na construção da rede bayesiana.

```
load('/home/gciotto/workspace/UNICAMP - EA072 - EFC3/Redes
bayesianas/projeto/data/dados_BN_EFC3.mat');

n_atributos = length(data(1,:));
n_dados = length(data);

maxs = zeros (1, n_atributos);
mins = zeros (1, n_atributos);

disp(sprintf('Numero de atributos eh %d e de registros eh %d', ...
      n_atributos, n_dados));

for k = 1:n_atributos

      maxs(k) = max (data(:,k));
      mins(k) = min (data(:,k));

      disp(sprintf('Maximo do atributo %d eh %d e minimo eh %d', ...
      k , maxs(k), mins(k)));
```

```
      for n=1:maxs(k)
      disp (sprintf('Elemento %d aparece %d vezes.', n, ...
            length(find(data(:,k) == n) )));
      end

end
```

Obtém-se, portanto, que cada um dos **10000** registros possuem **8** atributos. Somente o atributo 2 não e *booleano*, isto é, não apresenta somente dois valores. Tal atributo pode assumir 7 valores distintos. O número de aparições de cada valor de cada atributo, representado pela função *Q(x)*, está representado na tabela abaixo:

Tabela 2: número de aparições de cada elemento nos atributos.

Atributo	Q(1)	Q(2)	Q(3)	Q(4)	Q(5)	Q(6)	Q(7)
1	5014	4986	-	-	-	-	-
2	1377	1487	1355	1317	1587	1416	1461
3	5123	4877	-	-	-	-	-
4	7278	2722	-	-	-	-	-
5	1481	8519	-	-	-	-	-
6	1988	8012	-	-	-	-	-
7	5028	4972	-	-	-	-	-
8	4820	5180	-	-	-	-	-

2. A execução com os dados descritos acima gera a seguinte imagem:

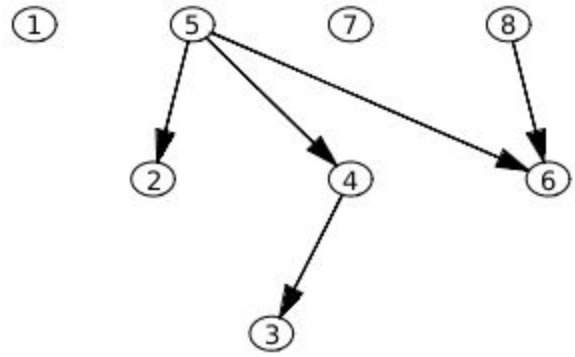

Figura 1: Rede bayesiana gerada pelo programa.

Nota-se que as variáveis 1 e 7 são independentes às demais e que as variáveis 5 e 8 são determinam as restantes. As probabilidades calculadas pelo programa estão mostradas nas tabelas abaixo.

Tabela 3: Propriedades para variável 1

Variável 1	
$P(V_1 = 1)$	0.5014
$P(V_1 = 2)$	0.4986

Tabela 4: Propriedades para variável 5

Variável 5	
$P(V_5 = 1)$	0.1481
$P(V_5 = 2)$	0.8519

Tabela 5: Propriedades para variável 7

Variável 7	
$P(V_7 = 1)$	0.5028
$P(V_7 = 2)$	0.4972

Tabela 6: Propriedades para variável 8

Variável 8	
$P(V_8 = 1)$	0.4820
$P(V_8 = 2)$	0.5180

Tabela 7: Propriedades para variável 2

Variável 2							
x	1	2	3	4	5	6	7
$P(V_2 = x \mid V_5 = 1)$	0.2633	0.1904	0.1769	0.1641	0.1182	0.0655	0.0216
$P(V_2 = x \mid V_5 = 2)$	0.1159	0.1414	0.1283	0.1261	0.1657	0.1548	0.1677

Tabela 8: Propriedades para variável 3

Variável 3		
x	1	2
$P(V_3 = x \mid V_4 = 1)$	0.3373	0.6627
$P(V_3 = x \mid V_4 = 2)$	0.9802	0.0198

Tabela 9: Propriedades para variável 4

Variável 4		
x	1	2
$P(V_4 = x \mid V_5 = 1)$	0.5949	0.4051
$P(V_4 = x \mid V_5 = 2)$	0.7509	0.2491

Tabela 10: Propriedades para variável 6

Variável 6		
x	1	2
$P(V_6 = x \mid V_5 = 1\ V_8 = 1)$	0.8788	0.1212
$P(V_6 = x \mid V_5 = 1\ V_8 = 2)$	0.5888	0.4112
$P(V_6 = x \mid V_5 = 2\ V_8 = 1)$	0.0528	0.9472
$P(V_6 = x \mid V_5 = 2\ V_8 = 2)$	0.1587	0.8413

3. As variáveis 1 e 7 podem ter sido adicionadas posteriormente aos dados e, portanto, são independentes às demais.

4. De acordo com a tabela 6, $P(V_8 = 2) = 0.5180$.

5. De acordo com a tabela 10, $P(V_6 = 2 \mid V_5 = 1\ V_8 = 2) = 0.4112$.

6. Utilizando a fórmula de Bayes, apresentada na equação (4), obtém o resultado (5).

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{4}$$

Logo, obtém-se

$$P(V_5 = 2|V_3 = 2) = \frac{P(V_3=2|V_5=2)P(V_5=2)}{P(V_3=2)} \tag{5}$$

$P(V_5 = 2)$ é dado pela tabela 4 e vale $P(V_5 = 2) = 0.8519$.

$P(V_3 = 2)$ é dado por

$$P(V_3 = 2) = P(V_3 = 2|V_4 = 1) * P(V_4 = 1) + P(V_3 = 2|V_4 = 2) * P(V_4 = 2) \tag{6}$$

Sendo que

$$P(V_4 = 1) = P(V_4 = 1|V_5 = 1)P(V_5 = 1) + P(V_4 = 1|V_5 = 2)P(V_5 = 2) = \\ = 0.5949 * 0.1481 + 0.7509 * 0.8519 = 0.7277964$$

e

$$P(V_4 = 2) = 1 - P(V_4 = 1) = 0.2722036$$

Enfim,

$$P(V_3 = 2) = 0.6672 * 0.7277964 + 0.0198 * 0.2722036 = 0.490975389$$

O último passo é, portanto:

$$P(V_3 = 2|V_5 = 2) = P(\{V_3 = 2|V_4 = 1\} \cup \{V_3 = 2|V_4 = 2\}|V_5 = 2) = \\ = 0.85 * (0.6627 * 0.7277964 + 0.0198 * 0.2722036) = 0.4145453$$

Assim, a equação (5) se torna

$$P(V_5 = 2|V_3 = 2) = \frac{0.4145453 * 0.8519}{0.490975389} = 0.719285$$

4. TensorFlow

O *TensorFlow* foi criado originalmente pela Google por pesquisadores trabalhando no Google Brain Team, na area de inteligência de máquina, aprendizado de máquina e redes neurais, mas o sistema era abrangente o suficiente para ser aplicado em diferentes areas. Ele é usado pela Google em várias áreas, entre elas: reconhecimento de discurso, Google Photo, Gmail.

Este software é uma biblioteca open source, voltada para computação numérica usando os chamados de Data Flow Graph. Estes gráficos, descrevem computações matemáticas com grafos direcionados com nós e arestas. Os nós tipicamente representam operações matemáticas mas podem ser *end points* usados para leitura de dados. As arestas descrevem a relação de input/output entre os nós. O fluxo de tensores pelo gráfico é da onde o nome se originou, *TensorFlow*.

TensorFlow pode ser baixado diretamente do site (https://www.tensorflow.org/get_started/os_setup.html#download-and-setup), e instalado como uma biblioteca em Python, disponivel para Ubuntu, Windows e MAC OS X. Também existem exemplos no site de como criar seu primeiro programa e alguns tutorias.

```
$ python

>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> print sess.run(hello)
Hello, TensorFlow!
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> print sess.run(a+b)
42
>>>
```

Figura 2: TensorFlow Hello World.

Os principais features do *TensorFlow* são grande flexibilidade, portabilidade, auto-diferenciação, diferentes opções de linguagem de programação (Python e C++), maximiza a performace, conecta produtores a pesquisadores.

Computações matemáticas são facilitadas e otimizadas com o uso do *TensorFlow*, segundo os criadores, este software pode fazer o computador enxergar e entender uma imagem. Ele deve trazer uma nova compreensão em problemas existentes o que pode possibilitar novos produtos.

5. Referências

- Davis, Goadrich, *"The Relationship Between Precision-Recall and ROC Curves"*.
- http://storm.cis.fordham.edu/~gweiss/data-mining/datasets.html. Acesso em 29/11.
- https://www.tensorflow.org/get_started/os_setup.html#download-and-setup. Acesso em 29/11.