Mestrado em Inteligência Artificial e Sistemas Inteligentes

Extracção de Conhecimento de Dados I

Trabalho Modelos Múltiplos

11 de Janeiro de 2005

António Jesus Monteiro de Castro

Aluno 040594004

Faculdade de Engenharia da Universidade do Porto

ÍNDICE

Objectivo e Enunciado do Problema

O objectivo deste trabalho é seleccionar um classificador a partir do conjunto de treino fornecido e, com o modelo obtido, classificar os exemplos do conjunto de teste que também foram fornecidos previamente.
De acordo com o pedido a metodologia deverá ser a seguinte:

1. Estimar a taxa de erro de dois classificadores, utilizando um método de amostragem. Os classificadores deverão ser uma árvore de decisão e um discriminante linear.
2. Escolher um dos métodos estudados na aula de modelos múltiplos para utilizar neste problema.
3. Comparar a taxa de erro entre os diversos classificadores, incluindo o modelo múltiplo.
4. Escolher um dos classificadores utilizados (podem ser um dos modelos base ou o modelo múltiplo) para classificar os exemplos dados.

Preparação dos Dados

A ferramenta escolhida para fazer este trabalho foi o WEKA. Para isso foi necessário preparar ambos os ficheiros de forma a que pudessem ser utilizados pelo programa. Assim, os ficheiros recebidos foram convertidos para o formato WEKA, bastando para isso, acrescentar os seguintes dados ao início de cada ficheiro:

```
@relation conjunto_treino

@attribute atr1 real
@attribute atr2 real
@attribute atr3 real
@attribute atr4 real
@attribute atr5 real
@attribute atr6 real
@attribute atr7 real
@attribute atr8 real
@attribute atr9 real
@attribute atr10 real
@attribute atr11 real
@attribute atr12 real
@attribute atr13 real
@attribute atr14 real
@attribute classe {0, 1}

@data
0,24.5,0.5,2,11,8,1.5,1,0,0,0,2,280,825,1
```

Relativamente ao ficheiro de teste a única diferença é o nome que em vez de *conjunto_treino* passo a ser *conjunto_teste* e o facto de nos dados não termos o último atributo (aparece assinalado com ?).

Análise Exploratória dos Dados

Antes de se avançar para a utilização dos classificadores e do calculo das respectivas taxas de erro, convém conhecer um pouco melhor os dados que foram disponibilizados, pois isso pode obrigar a ter que fazer algumas alterações antes da utilização dos classificadores.

Alguns dados estatísticos

Em primeiro lugar posso verificar que o conjunto de treino enviado tem 15 atributos, sendo o último correspondente à classe. Assim, as classes podem ter dois valores: 0 (zero) e 1 (um).

A classe 0 classifica 344 observações (aparece a azul nos gráficos) e a classe 1 classifica 276 observações (aparece a vermelho nos gráficos).

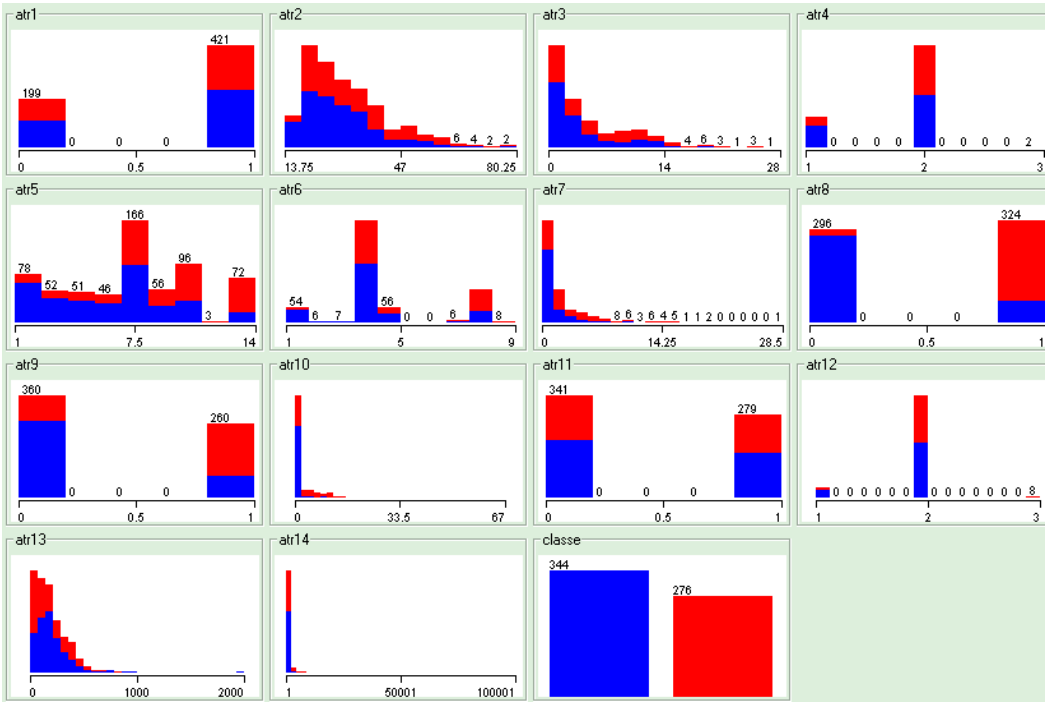
Relativamente aos outros atributos optei por resumir os dados relevantes na tabela seguinte:

Atr	*Tipo*	*NA*	*Distinct*	*Unique*	*Min*	*Max*	*Média*	*D.P.*
1	Num.	0	2	0	0	1	0.679	0.467
2	Num.	0	333	175	13.75	80.25	31.496	11.705
3	Num.	0	201	97	0	28	4.741	4.965
4	Num.	0	3	0	1	3	1.774	0.426
5	Num.	0	14	0	1	14	7.394	3.709
6	Num.	0	8	0	1	9	4.653	1.99
7	Num.	0	121	53	0	28.5	2.212	3.376
8	Num.	0	2	0	0	1	0.523	0.5
9	Num.	0	2	0	0	1	0.419	0.494
10	Num.	0	23	5	0	67	2.408	4.99
11	Num.	0	2	0	0	1	0.45	0.498
12	Num.	0	3	0	1	3	1.924	0.31
13	Num.	0	160	103	0	2000	185.837	172.126
14	Num.	0	217	171	1	100001	1053.506	5442.141

Significado das colunas:

Atr: O nome do atributo. Neste caso não têm nenhum signficado em especial a não ser especificar a sua ordem.
Tipo: Se é numérico (num.) ou nominal.
NA: Indica quantos valores omissos é que o atributo tem.
Distinct: O número de diferentes valores que os dados contêm para este atributo.
Unique: O número de observações nos dados que têm um valor para este atributo que mais nenhuma outra observação tenha.
Min: O valor mínimo do atributo.
Max: O valor máximo do atributo.
Média: O valor médio do atributo.
D.P.: O desvio padrão.

Os histogramas seguintes também nos permitem verificar a distribuição dos valores de cada atributo pelo seu domínio bem como o seu impacto na classificação.



Valores Omissos

Esta verificação é muito importante uma vez que a existência de valores omissos não tratados pode afectar o trabalho que se pretende realizar. Da tabela acima podemos verificar que não existem valores omissos no conjunto de dados enviados.

Se tivesse valores omissos poderíamos resolver a situação da seguinte forma:

1. Recorrendo à sua substituição pelo valor mais frequente desse atributo.
2. Substituir pelo valor mais frequente nos casos de treino mais semelhantes.
3. Utilizar técnicas mais sofisticadas, tais como, utilizar o atributo mais correlacionado com o atributo em falta (CART).

Por todas estas razões é usual, nos conjuntos de dados que têm valores omissos, fazer-se o estudo da correlação entre os vários atributos. Esse estudo permitiria fazer a melhor opção no que diz respeito à substituição desses valores.

Como não é o caso deste conjunto de dados optei por não fazer esse estudo.

Método de Amostragem e Classificadores Escolhidos

O conjunto de treino fornecido tem 620 observações. Nestas condições optei por utilizar *10-fold cross validation* porque é o método mais fiável quando estamos em presença de poucos registos. Se quisesse utilizar o método clássico de treino-teste (70/30) teria de repetir várias vezes (normalmente 10) de forma a poder calcular uma média dos erros. Caso o número de observações fosse substancialmente maior (por exemplo, acima de 15000), então poderia ter usado treino-teste (70/30) só com uma execução.

Relativamente aos classificadores base escolhi os seguinte:

- Logistic como discriminante linear.
- J48 como árvore de decisão.

Relativamente ao modelo múltiplo e apesar de ser pedido para utilizar só um, resolvi experimentar os três mais significativos, são eles:

- Bagging.
- Adaboost
- Stacking

Método de Trabalho

A metodologia que vou seguir para escolher o melhor classificador e, assim, classificar os dados de teste é a seguinte:

1. Calcular a taxa de erro para os classificadores base e ordená-los de acordo com a menor taxa de erro.
2. Utilizar o modelo *Bagging* usando como classificador base o que tiver menor erro.
3. Utilizar o modelo *Adaboost* usando como classificador base o que tiver menor erro.
4. Utilizar o modelo *Stacking* usando como classificadores base o *Logistic* e o *J48* e como classificador de nível 1 o melhor de ambos.
5. Ordenar as cinco taxas de erro e utilizar o classificador com menor taxa de erro para classificar os dados de teste.
6. Finalmente, analisar os resultados e preparar o ficheiro com as previsões obtidas.

Taxa de erro dos classificadores base

Usando o WEKA com os valores por defeito para cada classificador, obtive os seguintes resultados:

Logistic

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

Correctly Classified Instances         539               86.9355 %
Incorrectly Classified Instances        81               13.0645 %
Kappa statistic                          0.7371
K&B Relative Info Score              38873.7756 %
K&B Information Score                  385.5125 bits      0.6218
bits/instance
Class complexity | order 0             614.6358 bits      0.9913
bits/instance
Class complexity | scheme              308.4152 bits      0.4974
bits/instance
Complexity improvement     (Sf)        306.2206 bits      0.4939
bits/instance
Mean absolute error                      0.1956
Root mean squared error                  0.3195
Relative absolute error                 39.596  %
Root relative squared error             64.291  %
Total Number of Instances              620

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0.858     0.116      0.902     0.858     0.879    0
  0.884     0.142      0.833     0.884     0.858    1

=== Confusion Matrix ===

   a   b   <-- classified as
 295  49 |   a = 0
  32 244 |   b = 1
```

J48

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

Correctly Classified Instances         524               84.5161 %
Incorrectly Classified Instances        96               15.4839 %
Kappa statistic                          0.6857
K&B Relative Info Score              39376.0048 %
K&B Information Score                  390.4931 bits      0.6298
bits/instance
Class complexity | order 0             614.6358 bits      0.9913
bits/instance
Class complexity | scheme            20697.9566 bits     33.3838
bits/instance
Complexity improvement     (Sf)     -20083.3208 bits    -32.3925
bits/instance
Mean absolute error                      0.188
Root mean squared error                  0.3575
```

```
Relative absolute error                 38.0509 %
Root relative squared error             71.9302 %
Total Number of Instances              620

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0.872     0.188      0.852     0.872     0.862    0
  0.812     0.128      0.836     0.812     0.824    1

=== Confusion Matrix ===

   a   b   <-- classified as
 300  44 |   a = 0
  52 224 |   b = 1
```

Ordenando os dados obtidos fica:

Classif.	*MAE*	*RMSE*	*RAE*	*RRSE*	*Tx. Acerto*	*Tx. Erro*
Logistic	0.1956	0.3195	39.596%	64.291%	86.9355%	13.0645%
J48	0.188	0.3575	38.0509%	71.9302%	84.5161%	15.4839%

Taxa de erro dos modelos múltiplos

A escolha dos classificadores a utilizar em modelos múltiplos deve ter como critério principal o facto de eles não errarem ambos nas mesmas previsões, ou seja, devem ter erros não correlacionados. *Citando [Gama99] no capítulo 6.6.1 "As it was expected the lowest degree of correlation is between decision trees and Bayes and between decision trees and discrim" e como estamos a usar uma árvore de decisão e um discriminante linear, assumo que a taxa de erro correlacionado deste dois classificadores será adequada para a sua utilização num modelo múltiplo. Logicamente que, para ser rigoroso, o que eu devia fazer era calcular o erro correlacionado com os dados em questão*.
Uma vez que o *Logistic* foi o que obteve menor taxa de erro, vou usá-lo como classificador base no *Bagging* e, também, no *Adaboost*. O Resultado obtido foi:

Bagging

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

Correctly Classified Instances         537               86.6129 %
Incorrectly Classified Instances        83               13.3871 %
Kappa statistic                          0.7304
K&B Relative Info Score              38998.658  %
K&B Information Score                  386.751  bits      0.6238
bits/instance
Class complexity | order 0            614.6358 bits      0.9913
bits/instance
Class complexity | scheme             307.749  bits      0.4964
bits/instance
Complexity improvement     (Sf)       306.8868 bits      0.495
bits/instance
Mean absolute error                      0.1942
Root mean squared error                  0.3188
Relative absolute error                 39.3167 %
Root relative squared error             64.1447 %
```

```
Total Number of Instances              620

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0.858     0.123      0.897     0.858     0.877    0
  0.877     0.142      0.832     0.877     0.854    1

=== Confusion Matrix ===

   a   b   <-- classified as
 295  49 |   a = 0
  34 242 |   b = 1
```

Adaboost

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

Correctly Classified Instances         539               86.9355 %
Incorrectly Classified Instances        81               13.0645 %
Kappa statistic                          0.7371
K&B Relative Info Score              39702.0693 %
K&B Information Score                  393.7267 bits      0.635
bits/instance
Class complexity | order 0             614.6358 bits      0.9913
bits/instance
Class complexity | scheme              332.9648 bits      0.537
bits/instance
Complexity improvement     (Sf)        281.671  bits      0.4543
bits/instance
Mean absolute error                      0.1911
Root mean squared error                  0.3304
Relative absolute error                 38.6805 %
Root relative squared error             66.4815 %
Total Number of Instances              620

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0.858     0.116      0.902     0.858     0.879    0
  0.884     0.142      0.833     0.884     0.858    1

=== Confusion Matrix ===

   a   b   <-- classified as
 295  49 |   a = 0
  32 244 |   b = 1
```

Para o *Stacking* utilizarei como classificadores base o *J48* e o *Logistic* e como classificador de nível 1 o melhor deles que, neste caso, foi o *Logistic*. O resultado obtido foi:

Stacking

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

Correctly Classified Instances         535               86.2903 %
Incorrectly Classified Instances        85               13.7097 %
```

```
Kappa statistic                          0.7234
K&B Relative Info Score              38564.8605 %
K&B Information Score                  382.449  bits      0.6169
bits/instance
Class complexity | order 0             614.6358 bits      0.9913
bits/instance
Class complexity | scheme              304.7758 bits      0.4916
bits/instance
Complexity improvement     (Sf)        309.86   bits      0.4998
bits/instance
Mean absolute error                      0.2004
Root mean squared error                  0.32
Relative absolute error                 40.5613 %
Root relative squared error             64.3803 %
Total Number of Instances               620

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0.863     0.138      0.887     0.863     0.875    0
  0.862     0.137      0.835     0.862     0.848    1

=== Confusion Matrix ===

   a   b   <-- classified as
 297  47 |   a = 0
  38 238 |   b = 1
```

Resultados de todos os classificadores

Classif.	*MAE*	*RMSE*	*RAE*	*RRSE*	*Tx. Acerto*	*Tx. Erro*
Logistic	0.1956	0.3195	39.596%	64.291%	86.9355%	13.0645%
Adaboost	0.1911	0.3304	38.6805%	66.4814%	86.9355%	13.0645%
Bagging	0.1942	0.3188	39.3167%	64.1447%	86.6129%	13.3871%
Stacking	0.2004	0.32	40.5613%	64.3803%	86.2903%	13.7097%
J48	0.188	0.3575	38.0509%	71.9302%	84.5161%	15.4839%

Análise dos Resultados e Obtenção das Previsões

Como se pode verificar pelo quadro anterior, temos dois classificadores que obtiveram a mesma taxa de erro: *Logistic* e *Adaboost*.
Mesmo olhando para os outros indicadores verifica-se que nuns casos são melhores (ex: MAE Adaboost melhor que MAE Logistic) mas noutros é o inverso (Ex: RMSE Logistic melhor que RMSE Adaboost).
Sendo assim, resolvi utilizar o *Logistic* para efectuar as previsões nos dados de teste.
Mais uma vez socorri-me das opções do WEKA para o fazer e o resultado está no ficheiro ***prev_mm_antonio_castro_logistic.txt*** que envio como anexo a este trabalho.
Para além deste relatório e do ficheiro com as previsões, envio somente a título de curiosidade, um ficheiro com alguns comandos em R que cheguei a experimentar para este trabalho.

Referências

[BK99] Eric Bauer e Ron Kohavi. *An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting and Variants*. Machine Learning, 36:105-139, 1999.

[Gama99] João Gama. *Combining Classification Algorithms*. Tese de doutoramento. Faculdade de Ciências da Universidade do Porto, 1999.

[Gama04] João Gama. *Acetatos da Cadeira de Extracção de Conhecimento de Dados 1*. Faculdade de Economia da Universidade do Porto, 2004.

[Weka02] Richard Kirkby. *WEKA Explorer User Guide for Version 3-3-4*. Universidade de Waikato, 2002.

[WF99] Ian H. Witten e Eibe Frank. *Data Mining, practical machine learning tools and techniques with Java Implementations.* Morgan Kaufmann Publishers, 1999.