Projeto Aprendizado de Máquina

CTC- 17 Inteligência Artificial

Prof. Paulo André Castro

Shelly Leal

1. Objetivo

Implementar algoritmos de aprendizado de máquina utilizando-se um classificador baseado em árvore de decisão e um a priori. O objetivo é fazer a comparação entre os resultados utilizando-se uma base de dados de um conjunto de filmes, usuários e classificações, fornecida pelo grupo de pesquisa MovieLens da Universidade de Minnesota.

2. Desenvolvimento do Projeto

2.1 Descrição da Implementação

Os dois classificadores foram implementados em arquivos diferentes com a linguagem C++, utilizando-se o compilador GCC no Windows.

Inicialmente foi realizada a leitura dos arquivos users.dat, movies.dat e ratings.dat, os quais forneciam a base de dados do sistema de filmes.

O arquivo ratings.dat contia os dados no formato UserID:: MovieID :: Rating :: Timestamp, com a variação de 6040 usuários, 3952 filmes e classificação dos filmes até 5 estrelas.

A rotina ReadData() foi responsável pela leitura linha a linha das informações de cada arquivo e armazenamento em um respectivo vetor das estruturas de user, movies e ratings.

A partir disso os dados de cada estrutura foram unificados numa estrutra no formato UserID::MovieID::Rating::Timestamp, para unificar as informações respectivas equivalentes dos filmes. A classe responsável por esse trabalho se denomina MovieBlock().
A diferença entre os métodos dos arquivos arvoredecisao.c e priori.c foram os métodos dos classificadores utilizados respectivamente, DecisionTree() e AvgClassifier(), os quais serão descritos a seguir.

2.2 Classificador baseado em árvore de decisão

Os dados utilizados para gerar a árvore foram baseados nas informações de gênero, idade e ocupação que foram disponibilizados nos arquivos e armazenados, gerando os nós filhos a partir de cada classificação gerada.

Inicialmente é criado o nó raiz com as probabilidades de cada classe. Em seguida, para cada nó que não é folha, é necessário gerar divisões de acordo com as

possibilidades de cada um (por exemplo, a divisão entre idades dos usuários). A folha era encontrada quando não havia mais divisão possível e a probabilidade de classificação era 100%.

Quando é necessário consultar pela árvore de decisão, a função Search é utilizada para buscar o nó filho correspondente ao parâmetro de ramificação do nó pai.

2.2 Classificador a priori

A classificação a priori foi simplificada de forma a utilizar apenas a média truncada das classificações dos filmes.

Dessa forma foram realizadas as somas de cada classificações e realizada a respectiva média. O default para as classificações antes de calcular as médias era o valor 3 (entre 1 e 5).

3. Resultados Obtidos

Os testes foram realizados comparando-se as taxas de acerto e o erro médio para os casos de validação e de treinamento dos dois classificadores.

Foi decidido comparar para um conjunto de 50 e um de 100 filmes aleatórios no total. Para isso, as estrutras de dados de usuários, filmes e classificações foram aleatoriamente trocadas e algumas das posições foram selecionadas dentro de uma nova estrutura menor de um conjunto de filmes.

Os resultados foram demonstrados a seguir:

```
C:\MinGW\bin>arvoredecisao.exe
Feita a leitura dos arquivos

Reiniciando a arvore com 50 blocos aleatorios
Construindo a arvore...

Conjunto de treino com 1000159 blocos:
Erro medio: 0.000 Taxa de acerto: 100.00%

Conjunto de validacao com 50 blocos:
Erro medio: 1.058 Taxa de acerto: 36.00%


Reiniciando a arvore com 100 blocos aleatorios
Construindo a arvore...

Conjunto de treino com 1000109 blocos:
Erro medio: 0.000 Taxa de acerto: 100.00%

Conjunto de validacao com 100 blocos:
Erro medio: 1.342 Taxa de acerto: 27.00%
```

```
C:\MinGW\bin>priori.exe
Feita a leitura dos arquivos

Reiniciando a arvore com 50 blocos aleatorios
Fazendo a classificacao a priori...
Conjunto de treino com 1000159 blocos:
Erro medio: 1.067 Taxa de acerto: 37.71%

Conjunto de validacao com 50 blocos:
Erro medio: 0.949 Taxa de acerto: 44.00%


Reiniciando a arvore com 100 blocos aleatorios
Fazendo a classificacao a priori...
Conjunto de treino com 1000109 blocos:
Erro medio: 1.067 Taxa de acerto: 37.71%

Conjunto de validacao com 100 blocos:
Erro medio: 1.109 Taxa de acerto: 41.00%
```

Os resultados podem ser resumidos na tabela a seguir: (Erro acima e taxa de acerto abaixo)

Total de filmes	Classificador			
	Àrvore de Decisão		Priori	
	Treino	Validação	Treino	Validação
50	0 100%	1.058 36%	1.067 37.71%	0.949 44%
100	0 100%	1.342 27%	1.067 37.71%	1.109 41%

Pode se notar que para os casos de treinamento, a árvore de decisão apresentou um resultado equivalente ao ideal, o que torna um classificador preferível em relação ao a priori, o qual, devido à sua simplicidade, apresentou uma taxa de acerto relativamente menor. O fato de utilizar apenas a média para a classificação tornou as aproximações maiores e a maior chance de gerar erros na hora de haver uma consulta.
Porém, quando são analisados os casos de validação, o classificador a priori se tornou muito mais eficiente, ou seja, a árvore de decisão foi menos confiável.

4. Conclusões

O classificador a priori apresentou-se vantajoso pela simplicidade de implementação, porém, em casos de treinamento, foi demonstrado que a árvore de decisão é recomendada em sentido de confiabilidade. Pare resolver esse

problema da mesma, poderia ser estudado formas de simplificações dela, como por exemplo, foi estudado sobre o overfitting, o qual permite a redução de informações desnecessárias para a geração da árvore.
O trabalho foi interessante por permitir observar essa comparação e foi mais simplificado obter a implementação visto à facilidade de acesso aos materiais sobre árvore de decisão.
Seria interessante obter mais materias relativos à Naive Bayes, o que não foi muito encontrado especificamente e poderia ser explorado separadamente num próximo projeto.
No geral, a aula de aprendizado de máquina foi bem explorada dentro do projeto.