Instituto Tecnológico de Aeronáutica

Departamento de Engenharia de Software
Divisão de Ciência da Computação
Laboratório de CTC-17

Relatório da atividade

Projeto 3:
Aprendizado de máquina e árvore de decisão

Lucas França de Oliveira, lucas.fra.oli18@gmail.com
Professor: Prof. Paulo André Castro , pauloac@ita.br

10 de outubro de 2017

1 Introdução

Este trabalho consiste no estudo e aplicação de aprendizado de máquina supervisionado. O objetivo é treinar uma inteligência artificial para fazer a previsão de classificação de filmes a partir de uma base de dados com diversos exemplos de classificações por usuários reais. Os dados foram retirados de uma empresa real de gerência de filmes e há mais de um milhão de exemplos.

O aprendizado de máquina supervisioando consiste em criar um agente capaz de aprender com dados reais fornecidos, em que cada dado consiste em um conjunto de parâmetros e uma resposta. O objetivo do agente é, por meio da observação dos diversos exemplos, reconhecer um padrão entre os resultados e os parâmetros de forma a que ele possa reproduzi-lo.

Os códigos podem ser visualizados no repositório: https://github.com/splucs/Lab3_CTC17

2 Descrição do programa

Nesta atividade, foram fornecidos 3 arquivos: movies.dat, users.dat e ratings.dat. O primeiro contém dados de 3883 filmes no formato MovieID::Title::Genres. O segundo contém dados de 6040 usuários cadastrados no formato UserID::Gender::Age::Occupation::Zip-code. O terceiro contém dados de 1000209 avaliações no formato UserID::MovieID::Rating::Timestamp. Nos arquivos, os dados são dados um por linha. Inicialmente foram feitas 3 rotinas $ReadMovies$, $ReadUsers$ e $ReadRatings$ para fazer as leituras junto com o parsing das informações. Os dados são armazenados, respectivamente, em contâineres de std::vector das structs *movie*, *user* e *rating*.

Para se gerar uma base de dados que unifique as informações dos 3 três arquivos, foi criada uma struct *block* que guarda seis inteiros que representam o formato MovieID::Age::Gender::Occupation::Zip-code::Timestamp, nesta ordem. Foi codificada uma função $GenerateDataBlock$ que faz o cruzamento das tabelas para gerar os casos de teste. Este cruzamento é feito tomando cada rating e construindo o bloco com a struct movie e user equivalente. Para isso, foram desenvolvidos dois métodos: $GetMovieById$ e $GetUserById$. Eles recuperam a estrutura de filme e usuário, respectivamente, pelo identificador usando busca binária em $O(logn)$.

Neste processo, alguns dados sofrem uma simplificação. Esta serve para reduzir os problemas de $overfitting$. Caso a árvore tente representar os dados com muita precisão utilizando toda a informação detalhada do banco de dados, pode ocorrer este problema. Um modelo muito simples do espaço de dados gera $underfitting$, que faz o algoritmo errar pois não consegue distinguir um padrão claro nos dados. Um modelo muito complicado, por outro lado, detecta, além do padrão ótimo, o ruído dos dados, fazendo o algoritmo errar para casos novos, o chamado $overfitting$.

As reduções foram:

- O Zip-code (código postal) é reduzido ao seu prefixo, o que faz ele representar uma cidade em vez um punico bairro.
- A idade foi divida por 5 e truncada, o que faz os blocos se agruparem em grupos de 0-4 anos, 5-9, e assim por diante.
- O timestamp foi dividido por 10000000 e truncado, fazendo todo o banco de dados se agrupar em 3 grupos.

Com o intuito de se gerar casos de teste, os 1000209 dados foram separados aleatoriamente em dois grupos: o conjunto de treinamento e o conjunto de validação. O primeiro foi usado para fazer o treinamento do agente. O segundo, para a condução de testes. A proporção do tamanho do conjunto de validação para o total será discutida nos resultados deste relatório.

Foram implementados dois métodos de aprendizado: um por árvore de decisão e um por média simples de todas as classificações de cada filme. Cada método foi executado contra ambos o conjunto de treinamento e o de validação e suas saídas foram comparadas com as saídas reais. Seja N o tamanho do conjunto de testes, x_i a saída do método para o i-ésimo de teste e y_i a resposta real desse fornecida no banco de dados. A partir dessas comparações, foi calculado a taxa de acerto e o desvio padrão dado pela fórmula:

$$\sigma = \sqrt{\sum_{i=1}^{N} \frac{(x_i - y_i)^2}{N}} \tag{1}$$

2.1 Método de aprendizado: Árvore de decisão

Este método consiste na construção de uma estrutura de dados denominada "árvore de decisão". Ela organiza os casos de teste fazendo, a cada nível, uma partição segundo algum de seus parâmetros. A sua altura é, portanto, o número de parâmetros mais um. Este é um método famoso na literatura e com diversas aplicações na indústria. Ele á capaz de resolver consultas em tempo rápido. Ele não suporta atualização - adição de novos casos de treinamento -, mas esta não foi implementada por motivos de simplicidade no laboratório.

Por exemplo: a raíz separa o conjunto em partições segundo cada possível valor do primeiro parâmetro. Seus filhos separam cada conjunto dessa partição pelo segundo parâmetro, e assim por diante. O problema de classificação se reduz à busca numa árvore e a complexidade é de $O(p)$, em que p é o número de parâmetros, que neste problema é 6. A busca se inicia na raíz e termina sempre numa folha. No código, a função que implementa a construção da árvore é $BuildDecisionTree$.

As condições de parada na construção da árvore são:

- Não há casos de treino: a resposta é o valor mais comum do nó pai.
- Todos os casos de treino que o nó representa têm uma mesma resposta: esta é a resposta.
- Não há parâmetros não usados para se fazer partição: a resposta é o valor mais comum.
- Um novo parâmetro desconhecido ou um valor que não estava presente nos casos de treino foi encontrado: a resposta é o valor mais comum.

Para computar o valor da maioria de um conjunto de casos de treino, foi implementada uma função $GetMajorityValue$. O valor de maioria do pai da raíz é 3, um valor que tenta minimizar o desvio padrão considerando que qualquer classificação pode acontecer com igual probabilidade.

Para realizar uma consulta, basta começar pela raíz. Para cada nó, verifica-se qual parâmetro ele usa para fazer partição e prosseguir para o filho que representa o valor desse atributo no bloco de consulta. Ao chegar a uma folha, ela terá um valor e este é retornado. A função que executa esta tarefa é $Query$. Por fim, foi implementado um destrutor recursivo para a árvore $Delete$.

2.2 Método de aprendizado: Média das classificações

Este método é relativamente mais simples que o anterior e usa apenas as classificações de cada filme para o conjunto de treinamento. Cada filme tem um contador de classificações e um somador dessas. Para cada caso de treinamento, a classificação é somada no somador do filme equivalente e o contador deste é incrementado em um. Por fim, a média de cada filme é o somador desse divido pelo contador. Caso o contador seja 0, a média é definida como 3, pelo mesmo motivo dado acima. A função que executa esta tarefa é $ComputeMovieAverageRatings$.

3 Testes e Resultados

Para executar os testes, foi implementada a função *RunDataset*. Ele calcula o desvio padrão para cada método por um conjunto de casos fornecidos pelo argumento da função. Foi executada uma sequência de 10 testes para o problema. A diferença de cada um é a proporção de casos reservados para o conjunto de validação em relação ao total fornecido. O menor caso é 5% e o maior, 50%. Cada caso intermediário difere em 5%.

Para cada proporção, os dados foram completamente embaralhados e a árvore foi construída do zero. As médias do segundo método foram recalculadas e ambos os conjuntos, o de treinamento e o de validação, foram executados calculando-se seu desvio padrão.

O programa *decisiontree.cpp* pode ser acessado pelo anexo enviado ou pelo repositório no github. O código implementado em C++ utilizando o compilador GNU g++ 6.3 (-O3) em uma máquina 64 bits com processador Intel(R) Core(TM) i5-6300 CPU @ 2.40Hz apresentou a seguinte saída:

```
Reading movies... done: 3883 movies read
Reading ratings... done: 1000209 ratings read
Reading users... done: 6040 users read
Generating data blocks... done, number os blocks = 1000209
Separating 5.00% of data blocks for validation set... done, 50010 blocks
Building tree and computing average movie ratings... done, tree size 1098206
Testing training set (size 950199)... done, DT 0.195-98.53%, avg. 1.068-37.70%
Testing validation set (size 50010)... done, DT 1.252-35.22%, avg. 1.049-37.40%
Separating 10.00% of data blocks for validation set... done, 100020 blocks
Building tree and computing average movie ratings... done, tree size 1039885
Testing training set (size 900189)... done, DT 0.195-98.53%, avg. 1.024-37.95%
Testing validation set (size 100020)... done, DT 1.302-32.94%, avg. 1.403-35.24%
Separating 15.00% of data blocks for validation set... done, 150031 blocks
Building tree and computing average movie ratings... done, tree size 982091
Testing training set (size 850178)... done, DT 0.196-98.52%, avg. 1.034-37.81%
Testing validation set (size 150031)... done, DT 1.301-32.86%, avg. 1.253-36.77%
Separating 20.00% of data blocks for validation set... done, 200041 blocks
Building tree and computing average movie ratings... done, tree size 924940
Testing training set (size 800168)... done, DT 0.195-98.53%, avg. 1.033-37.78%
Testing validation set (size 200041)... done, DT 1.301-32.91%, avg. 1.216-36.85%
Separating 25.00% of data blocks for validation set... done, 250052 blocks
Building tree and computing average movie ratings... done, tree size 867912
Testing training set (size 750157)... done, DT 0.193-98.55%, avg. 1.026-37.86%
Testing validation set (size 250052)... done, DT 1.309-32.94%, avg. 1.205-36.61%
Separating 30.00% of data blocks for validation set... done, 300062 blocks
Building tree and computing average movie ratings... done, tree size 810643
Testing training set (size 700147)... done, DT 0.192-98.55%, avg. 1.020-38.00%
Testing validation set (size 300062)... done, DT 1.331-31.97%, avg. 1.204-35.89%
Separating 35.00% of data blocks for validation set... done, 350073 blocks
Building tree and computing average movie ratings... done, tree size 752590
Testing training set (size 650136)... done, DT 0.193-98.54%, avg. 1.016-38.06%
Testing validation set (size 350073)... done, DT 1.320-32.31%, avg. 1.192-35.80%
Separating 40.00% of data blocks for validation set... done, 400083 blocks
Building tree and computing average movie ratings... done, tree size 694612
Testing training set (size 600126)... done, DT 0.194-98.51%, avg. 1.015-38.04%
Testing validation set (size 400083)... done, DT 1.349-31.70%, avg. 1.206-35.43%
Separating 45.00% of data blocks for validation set... done, 450094 blocks
Building tree and computing average movie ratings... done, tree size 636582
Testing training set (size 550115)... done, DT 0.195-98.50%, avg. 1.012-38.09%
Testing validation set (size 450094)... done, DT 1.336-31.99%, avg. 1.200-35.54%
Separating 50.00% of data blocks for validation set... done, 500104 blocks
Building tree and computing average movie ratings... done, tree size 578647
Testing training set (size 500105)... done, DT 0.194-98.50%, avg. 1.010-38.11%
Testing validation set (size 500104)... done, DT 1.335-32.13%, avg. 1.202-34.79%
```

Os resultados, no formato "desvio padrão - taxa de acerto" em função da proporção do tamanho do conjunto de validação em relação ao total, foram resumidos na tabela:

Porcentagem de validação	árvore de decisão conj. de treino	média simples conj. de treino	árvore de decisão conj. de validação	média simples conj. de validação
5%	0.195-98.53%	1.068-37.70%	1.252-35.22%	1.049-37.40%
10%	0.195-98.53%	1.024-37.95%	1.302-32.94%	1.403-35.24%
15%	0.196-98.52%	1.034-37.81%	1.301-32.86%	1.253-36.77%
20%	0.195-98.53%	1.033-37.78%	1.301-32.91%	1.216-36.85%
25%	0.193-98.55%	1.026-37.86%	1.309-32.94%	1.205-36.61%
30%	0.192-98.55%	1.020-38.00%	1.331-31.97%	1.204-35.89%
35%	0.193-98.54%	1.016-38.06%	1.320-32.31%	1.192-35.80%
40%	0.194-98.51%	1.015-38.04%	1.349-31.70%	1.206-35.43%
45%	0.195-98.50%	1.012-38.09%	1.336-31.99%	1.200-35.54%
50%	0.194-98.50%	1.010-38.11%	1.335-32.13%	1.202-34.79%

O motivo de se variar o tamanho de conjunto de validação é a análise da probabilidade do algoritmo não conseguir distinguir um padrão em função do número de exemplos dados. Observa-se que, mesmo com 50% de casos reservados para validação, a tabela apresentou pouca variação. Um outro teste feito com até 90% dos casos reservados para validação mostrou um resultado semelhante. Este caso não foi inserido no relatório pois sua saída é muito grande.

No geral, observa-se que a árvore de decisão consegue representar o conjunto de treinamento de forma muito mais exata que por média simples. Teoricamente, a árvore de decisão seria capaz de representar este conjunto perfeitamente. O motivo de ocorrer o erro nos casos de treinamento são as reduções feitas de forma a se simplificar a modelagem e evitar $overfitting$. Foi realizado, também, um novo teste sem as simplificações, e o desvio padrão da árvore nesse conjunto caiu para 0.

Ao se mover para o conjunto de validação, o rendimento caiu e tornou-se pior que por média simples. O resultado foi semelhante no teste sem simplificação dos dados. Considerando que o classificador por média simples se saiu melhor, ele pode ser considerado com uma árvore de decisão de um parâmetro, um modelo muito mais simples que o adotado. Isso sinaliza para o problema de $overfitting$, e uma possível solução seria uma modelagem mais simples.

4 Conclusão

Todas as atividades foram concluídas com êxito. Com relação à simplificação de dados e à redução destes, determinou-se que não houve vantagem em fazê-la, pois a árvore de decisão teve um desempenho parecido com ou sem para os casos de validação. Para os casos de treino, a árvore de decisão os consegue representar plenamente se não houver simplificação.

O trabalho se mostrou útil para aprimorar o conhecimento sobre os métodos de inteligência artificial apresentados - aprendizado em máquina supervisionado e árvore de decisão. O problema estudado é interessante e remete a problemas comuns na indústria, sendo a execução deste laboratório um excelente exercício de aplicação das técnicas abordadas. O laboratório se mostrou simples de implementar, tendo o pseudo-código do algoritmo apresentado no material didátio do curso. Não houve impedimentos.