GradePrediction

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Inteligência Artificial

Autores:
João Diogo Trindade Guarda - 201303461
João Rafael de Figueiredo Cabral - 201304395
João Bernardo Martins de Sousa e Silva Mota - 201303462

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

30 de Março de 2017

1 Objectivo

O objetivo deste trabalho consiste na aplicação de Redes Neuronais Artificiais na predição do desempenho de estudantes. Pretende-se desenvolver uma solução que treine uma Rede Neuronal de forma supervisionada, com recurso ao método *Back-Propagation*. Para o efeito deve ser usado um conjunto de dados previamente obtidos. A solução desenvolvida deve, findo o processo de treino, permitir ao utilizador introduzir no sistema os dados de um aluno, os quais serão normalizados e fornecidos à rede neuronal. A saída do programa deverá ser a estimativa de nota calculada pela rede neuronal.

São fornecidos dois conjuntos de dados diferentes: um relativo às classificações da disciplina de Matemática e outro relativo às classificações da disciplina de Português.

2 Descrição

2.1 Especificação

O trabalho vai consistir no desenvolvimento de um programa de computador capaz de efetuar três operações fundamentais: pré-processar o *dataset* [1] fornecido por forma a possibilitar o seu processamento pela rede neuronal, treinar a rede neuronal com um conjunto de dados disponibilizados previamente e, por fim, possibilitar a aplicação da rede neuronal treinada à previsão de avaliações de alunos com base em novas entradas a serem fornecidas pelo utilizador final da aplicação. Estes três componentes do trabalho encadeiam-se de uma forma sequencial entre si, usando como entrada os dados de saída da etapa anterior. Para a implementação da rede neuronal vai ser usada uma biblioteca externa, a *Neuroph*, que disponibiliza uma série de métodos e classes que possibilitam a especificação, treino e aplicação de uma rede neuronal.

2.1.1 Dataset

Vai ser empregue um conjunto de dados de treino que contêm dados de diversos tipos, incluindo numéricos, binários e nominais. Os dados que são expressos em categorias já surgem adequadamente codificados com uma codificação do tipo *1-of-C*.

O dataset contém dados em escalas numéricas distintas, pelo que se torna necessário normaliza-los antes de os fornecer à rede neuronal, por forma a evitar problemas com valores cuja média seja muito diferente da dos restantes. Optou-se por normalizar os dados recorrendo ao método *MaxMin*, cuja implementação será fornecida pela biblioteca *Neuroph* anteriormente mencionada.

2.1.2 Estrutura da rede

Para a escolha da estrutura da rede neuronal optou-se por uma abordagem essencialmente experimental, começando com uma rede pequena e aumentando gradualmente o número de nós e *hidden layers* até que se deixasse de verificar um ganho apreciavel de velocidade de aprendizagem ou uma redução significativa do erro médio obtido na fase de testes. Uma possível "regra"para determinar o número de nós das *hidden layers*

baseia-se no facto de para um sistema de equações ser resolvido tem de ter mais equações linearmente independetes que incógnitas. Baseando-se nas seguintes igualdades:

$$\begin{cases} \text{n}^{\text{o}} \text{ equações} = \text{n}^{\text{o}} \text{ exemplos no } \textit{dataset} \\ \text{n}^{\text{o}} \text{ incógnitas} = \text{n}^{\text{o}} \text{ arrestas na rede} \end{cases}$$

Com base neste princípio para garantir que o sistema é resolvível utilizaremos 3 a 5 vezes mais equações que incógnitas de forma a minimizar a hipótese de existencia de equações linearmente dependentes. A escolha do número de nós deve ser cuidadosa visto que o uso de poucos nós pode impedir a rede de convergir.

Uma alternativa pode ser utilizar um número de nós entre o número de nós de entrada e de saída:

$$(inputs + outputs) * 2/3$$

Os nós de entrada serão alvo de um estudo cuidado de forma a perceber quais são os nós que poderão introduzir ruído na rede. No entanto, se todos os nós forem utilizados a rede vai ter 32 nós de entrada, por ser essa a quantidade de atributos do *dataset* e sempre 1 nó de saída, a classificação final do aluno, que se pretende prever.

2.1.3 Amostragem para treino e teste

Para treinar a rede neuronal e avaliar a sua competência, dividir-se-á o *dataset* em duas partes. A primeira será utilizada para treino e a segunda para teste. Estudar-se-á também quais as percentagens mais eficazes para o efeito. Sabendo que a maior parte das entradas será utilizada para treino o objectivo passa por perceber até que ponto é vantajoso inserir mais dados perdendo rapidez de treino mas ganhando precisão no modelo de previsão. Uma boa base de partida será utilizar cerca de 2/3 para treino e 1/3 para teste.

2.1.4 Treino

Para treinar a rede neuronal, ou seja ajustar os pesos das arestas ao problema que se pretende resolver, vai ser utilizado um mecanismo de aprendizagem supervisionada conhecido como *Back-Propagation*. Este mecanismo, cuja implementação é também fornecida pela biblioteca *Neuroph* funciona dando à rede um conjunto de casos de teste para cálculo, bem como o resultado esperado para esse mesmo conjunto de entradas.

A rede efetua a propagação dos valores pelos nós e compara o valor dos nós de saída com o valor esperado. Essa comparação permite o cálculo de um erro. Este calculo é posteriormente utilizado para atualizar os valores dos pesos das arestas. O processo é repetido até que o erro a classificar o conjunto de dados de treino seja menor do que um determinado valor, ou até que o máximo de iterações seja excedido. Importa ainda definir uma taxa de aprendizagem, que influenciará a velocidade a que os valores dos nós são atualizados. Valores mais pequenos levam a treinos mais precisos, mas valores maiores agilizam a velocidade de treino.

2.1.5 Previsão de notas

Vai ser implementada uma interface que permita ao utilizador introduzir os dados correspondentes a alunos cuja nota pretende prever. Estes dados deverão ser fornecidos à rede neuronal, após passarem pelo processo de normalização descrito anteriormente. Por fim a saída da rede neuronal deve ser fornecida ao utilizador numa forma desnormalizada, para que surja na escala de 0 a 20 habitual, e não na representação interna do sistema.

Um possível problema com a forma como o *dataset* está estruturado é a inclusão de um atributo correspondente à escola frequentada pelo aluno. No universo de alunos no qual o dataset se baseia existem apenas duas escolas, contudo o programa deverá funcionar para alunos de qualquer escola. A rede neuronal, contudo, foi treinada com um conjunto de casos de treino que não contempla essa possível diversidade, e cujos dados foram normalizados tendo em conta a existência de duas escolas. Estudar-se-á a possibilidade de excluir esse atributo do *dataset* e as consequências dessa exclusão ao nível da fiabilidade dos resultados.

2.2 Trabalho Efectuado

Optou-se por não prosseguir com a implementação da rede neuronal de raiz. A biblioteca *Neuroph*, já amplamente referida na secção 2, será utilizada para estruturar a rede neuronal e para levar a cabo o processo de treino. O trabalho até agora desenvolvido centra-se, assim sendo, no estudo e análise dos dados a utilizar, nos processos de normalização que virão a ser utilizados. Os resultados e decisões já tomadas podem vir a sofrer alterações significativas à medida que o grupo for explorando outros métodos de normalização e possivelmente outras variantes de *Back-Propagation*.

2.2.1 Normalização de dados

No *dataset* utilizado são disponibilizadas as seguintes características sobre os estudantes:

1. school - Escola (binário: 'GP' - Gabriel Pereira ou 'MS' - Mousinho da Silveira)
2. sex - Sexo (binário: 'F' - feminino ou 'M' - masculino)
3. age - Idade (numérico: de 15 a 22)
4. address - Tipo de morada (binário: 'U' - urbano ou 'R' - rural)
5. famsize - Tamanho da família (binário: 'LE3' - menor ou igual a 3 ou 'GT3' - maior que 3)
6. Pstatus - Estado de coabitação dos pais (binário: 'T' - vivem juntos ou 'A' - separados)
7. Medu - Grau de educação da mãe (numérico: 0 - nenhum, 1 - educação primária (4º ano), 2 - 5º ao 9º ano, 3 - educação secundária (12ºano) ou 4 - ensino superior)

8. Fedu - Grau de educação do pai (numérico: 0 - nenhum, 1 - educação primária (4º ano), 2 - 5º ao 9º ano, 3 - educação secundária (12ºano) ou 4 - ensino superior)
9. Mjob - Trabalho da mãe (nominal: professor 'teacher', área da saúde 'health', administração pública ou policiar 'services', doméstico 'at_home' ou outro 'other')
10. Fjob - Trabalho do pai (nominal: professor 'teacher', área da saúde 'health', administração pública ou policiar 'services', doméstico 'at_home' ou outro 'other')
11. reason - Razão de escolha da escola (nominal: perto de casa 'home', reputação da escola 'reputation', preferência de curso 'course' ou outro 'other')
12. guardian - Encarregado de Educação (nominal: mãe 'mother', pai 'father' ou outro 'other')
13. traveltime - Tempo de viagem casa-escola e escola-casa (numérico: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
14. studytime - Número de horas semanais de estudo (numérico: 1 - <2 horas, 2 - 2 a 5 horas, 3 - 5 a 10 horas, ou 4 - >10 horas)
15. failures - number of past class failures (numérico: n se 1<= n <3, senão 4)
16. schoolsup - Apoio educacional extra (binário: sim ou não)
17. famsup - Apoio educacional da família (binário: sim ou não)
18. paid - Aulas extra pagas (Matemática or Português) (binária: sim ou não)
19. activities - Atividades extra-curriculares (binário: sim ou não)
20. nursery - Frequentou o ensino pré-escolar (binário: sim ou não)
21. higher - Quer em cursar no ensino superior (binário: sim ou não)
22. internet - Acesso a internet em casa (binário: sim ou não)
23. romantic - Numa relação romântica (binário: sim ou não)
24. famrel - Qualidade das relações familiares (numérico: de 1 - muito mau a 5 - muito bom)
25. freetime - Tempo livre fora da escola (numérico: de 1 - muito baixo a 5 - muito alto)
26. goout - Frequência com que sai com os amigos (numérico: de 1 - muito baixo a 5 - muito alto)
27. Dalc - Consumo de alcool nos dias de trablho (numérico: de 1 - muito baixo a 5 - muito alto)

28. Walc - Consumo de alcool nos fim de semanas (numérico: de 1 - muito baixo a 5 - muito alto)

29. health - estado actual de saúde (numérico: de 1 - muito mau a 5 - muito bom)

30. absences - número de faltas dadas (numérico: de 0 a 93)

De forma a simplificar o trabalho de aprendizagem da rede neuronal todas as entradas não numéricas são convertidas em numéricas. No caso de característica ser binária utilizam-se 0 para uma opção e 1 para outra outra. No caso de ser nominal, é utilizada uma codificação do tipo *1-of-C*:

Mjob 'teacher' - 0; 'health' - 1; 'services' - 4; 'at_home' - 3; 'other'- 2;

Fjob 'teacher' - 0; 'health' - 1; 'services' - 4; 'at_home' - 3; 'other'- 2;

reason 'home'- 0; 'reputation' - 1; 'course' - 3; 'other' - 2;

guardian 'mother' - 0; 'father' - 1; 'other' - 2;

2.2.2 Implementação da Rede Neuronal

Aqui apresenta-se uma implementação de um rede neuronal com a biblioteca externa *Neuroph*:

```
GradePrediction(){
    DataSet data = DataSet.createFromFile("student-por.csv",32,1,";",true);
    MaxMinNormalizer normalizer = new MaxMinNormalizer();
    normalizer.normalize(data);
    SubSampling sampling = new SubSampling(60,40);
    List<DataSet> sets = sampling.sample(data);
    trainingSet = sets.get(0);
    testSet = sets.get(1);

    System.out.println("Data Set size:" + data.size());
    System.out.println("Training Set size:" + trainingSet.size());
    System.out.println("Test Set size:" + testSet.size());

    double connections = (double)trainingSet.size()/4.0;
    int nodes = (int)Math.round(connections/32.0);
    System.out.println("Hidden Nodes:" + nodes);


    network = new MultiLayerPerceptron(TransferFunctionType.SIGMOID,32,nodes,1);
    BackPropagation bp = new BackPropagation();
    bp.setNeuralNetwork(network);
    bp.setLearningRate(0.1);
    bp.setMaxError(0.001);
    bp.setMaxIterations(1000);
    network.learn(trainingSet, bp);
```

```
    System.out.println();
    System.out.println("######### TRAINING #########");
    System.out.println();

    System.out.println("Total Iterations: " + bp.getCurrentIteration());
    System.out.println("Total error: " + bp.getTotalNetworkError());
}
```

Neste exemplo optamos por utilizar 60% dos dados para treinar a rede, os restantes 40% para teste, 1 camada escondida, um erro máximo de 0.001, um máximo de iterações de 1000 e uma taxa de aprendizagem de 0.1. Para o cálculo do número de nós escondidos da rede utilizamos a regra de em que o número de exemplos de traino devem ser 3 a 5 vezes superior que o número de arestas.

2.3 Resultados esperados e forma de avaliação

Para testar a rede neuronal obtida pelo módulo de treino vai ser utilizada uma parte dos dados fornecidos pelo *dataset* utilizado. Assim, alguns dos dados serão retirados do conjunto utilizado para treino da rede neuronal e utilizados para efetuar testes de exatidão aos resultados calculados pela rede. O conjunto de dados de teste será fornecido à rede já treinada e os erros acumulados serão guardados, sendo posteriormente calculados a sua média e desvio padrão. Os resultados dos testes efetuados à rede obtida pelas decisões até agora tomadas são de aproximadamente 1 valor de erro médio, representando uma anomalia de 5% na classificação, com desvio padrão de aproximadamente 1.1.

3 Conclusões

A análise do trabalho até agora efetuado e dos resultados que dele se obtiveram permite concluir que a utilização de uma rede neuronal é relativamente eficaz para efetuar a previsão das notas, tendo os resultados preliminares indicado que a rede empregue, apesar das decisões de normalização e de configuração da mesma ainda não estarem completamente desenvolvidas, é capaz de prever com pouco erro acumulado as notas dos alunos que são usados para teste.

Importa realçar que os testes empregues utilizam dados vindos da mesma base de dados gerada a partir de entrevistas a alunos de duas escolas, e portanto recolhidos de populações cujas caracteristicas podem apresentar semelhanças que não se verificarão com alunos de, por exemplo, outras escolas ou meios geográficos.

4 Recursos

4.1 Software

- Neuroph API

- Java Development Kit

4.2 Bibliografia

[1] P. Cortez and A. Silva. Using data mining to predict secondary school student performance. In A. Brito and J. Teixeira Eds., editors, *Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008)*, pages 5–12. EUROSIS, 2008.