UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
ESCOLA DE CIÊNCIAS E TECNOLOGIA

Mapeamento e Agrupamento da Situação dos Discentes

Discente: Alysson Rafael Oliveira de Lima
Docente: Orivaldo Santana Jr

Natal, 16 de outubro de 2019

Introdução

Muitos alunos abandonam as disciplinas na metade do semestre, muitas vezes, após ver que o rendimento não está indo da maneira necessária. Para evitar isto pensou-se em uma forma de obter com antecedência de tempo satisfatória uma previsão da situação de aprovação de alunos da disciplina de Lógica de Programação - LoP, e com isso ser possível evitar uma desistência e principalmente uma reprovação.

Para isso, este experimento, propõe identificar as características padrões que possam indicar uma possível reprovação. Através da utilização do modelo de Machine Learn Self-Organizing Map - SOM, buscaremos mapear os alunos em grupos homogêneos para uma decisão de intervenção pedagógica.

Modelo de Rede Neural

O **SOM** é um modelo de rede neural de aprendizado não supervisionado. Na entrada da rede é suportado N dimensões, através de 2 camadas, gerando o mapeamento para um conjunto de neurônios de saída. A saída por sua vez representa as posições dos neurônios em relação aos seus vizinhos na forma de um mapa tipicamente bi-dimensional.

Alguns pontos interessantes desse modelo de rede neural e que facilita na obtenção dos resultados procurados são os seguintes:

- Agrupamento de padrões (Clusters)
- Redução de dimensionalidade
- Descoberta de correlações escondidas entre os dados (mineração de dados)
- Compressão de dados (Extração de características)
- Classificação

Processo de Geração de Dados

Os dados foram extraídos de uma base gerada na turma de Lógica de Programação, da Escola de Ciências e Tecnologia, da UFRN nos períodos de 2017.2 até 2019.1. Consiste em informações sobre a quantidade de questões submetidas em 21 semanas, além de fornecer a situação ao final do semestre do aluno. Ao todo, a base conta com 948 registros.

Foi necessária uma adaptação à base, visto que, a mesma representava a situação do aluno com um dado em formato textual(APROVADO, APROVADO POR NOTA, REPROVADO, REPROVADO POR NOTA e REPROVADO POR MÉDIA E POR FALTAS) sendo substituído por números para uso no modelo. Essa alteração foi possível com a utilização do seguinte código:

```
le = preprocessing.LabelEncoder()
dataset["situacao"] = le.fit_transform(dataset["situacao"])
```

Após o processo de transformação, os dados referentes à situação foram representados através de números, onde:

```
0 = APROVADO
1 = APROVADO POR NOTA
2 = REPROVADO
3 = REPROVADO POR NOTA
4 = REPROVADO POR MÉDIA E POR FALTAS
```

Com isso, é possível utilizar os algoritmos do modelo SOM para realizar a atividade proposta.

Desenvolvimento

No experimento foi utilizado todas as 21 colunas da base de dados que representam a quantidade de submissões em cada semana, além da coluna que representa a situação do aluno ao final do semestre.

```
#Pegando os dados necessários
X = dataset.iloc[:,2:22].values
y = dataset.iloc[:, 25].values
```

Para um melhor funcionamento do algoritmo SOM, após os dados serem separados foi realizado um processo de normalização dos valores com a utilização da função **MinMaxScaler** da biblioteca **sklearn.preprocessing** como mostrado a seguir:

```
# Normalizando
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
X = sc.fit_transform(X)
```

Após os dados serem padronizados, iniciou o processo de treinamento da rede. Após vários testes a fim de encontrar uma forma que facilitasse a identificação de grupos de características e um melhor agrupamento onde fosse possível visualizar pelo menos um grupo com 100% de probabilidade de reprovação, chegamos às seguintes configurações de treinamento da rede:

```
# Treinamento da SOM
from minisom import MiniSom
som = MiniSom(x = 10, y = 10, input_len = 20, sigma = 1.0,
learning_rate = 0.5)
som.random_weights_init(X)
som.train_random(data = X, num_iteration = 10000)
```

Para visualização dos dados pós treinamento foi criada uma matriz contendo o total de dados agrupados e outra com o total de aprovados através do seguinte código:

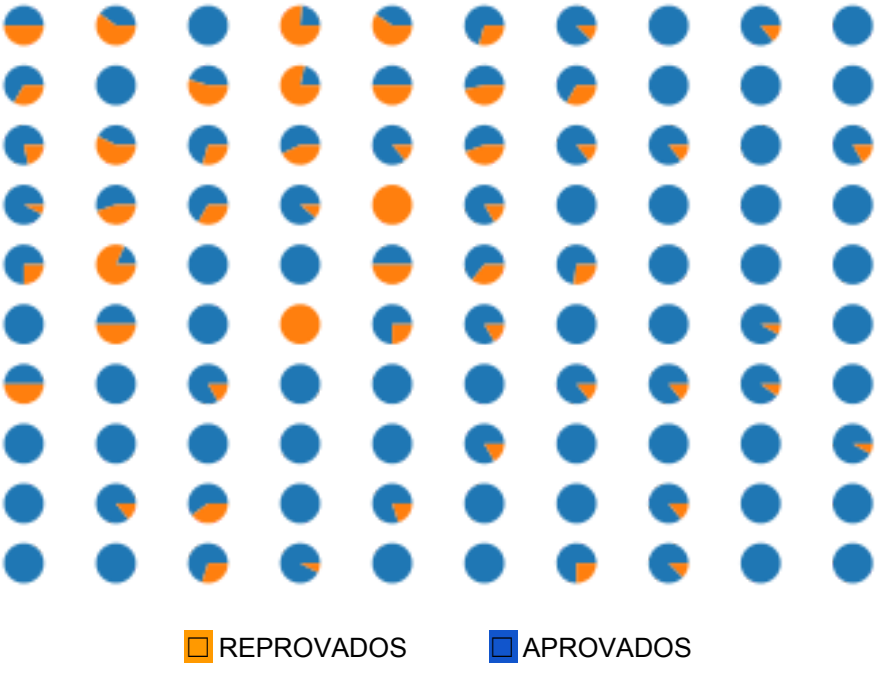
```
# matriz de zeros para o contador de reprovados
MContT = np.zeros((10,10))
# matriz de zeros para contador de aprovados
MContAp = np.zeros((10,10))

cont = 0;
for x in X:
  pos = som.winner(x)
  if (y[cont] < 2): #Aprovado
    MContAp[pos] += 1
  MContT[pos] += 1
  cont= cont+1

print("Total:")
print(MContT)

print("Aprovados")
print(MContAp)
```

Gerando as seguintes matrizes:

```
Total:
[[ 10.   5.   1. 161.  32.  17.   8.   7.   7.   6.]
```

```
 [  6.   1.  22.  14.  12.  15.   6.   2.   3.   3.]
 [ 14.  21.  17.  14.  29.   9.   7.   7.   5.   6.]
 [ 13.  29.   3.   9.   1.   6.   2.   2.   3.   2.]
 [ 24.  11.   1.   6.   2.  14.  11.  11.   5.   6.]
 [  5.  10.   4.   2.   4.   6.   2.   2.  12.   4.]
 [ 12.   3.   6.   5.   3.   4.   7.   7.  10.   4.]
 [  2.   4.   6.   5.   7.   6.   7.   6.  18.  12.]
 [ 13.   7.   5.   5.   5.   5.   6.   7.  12.   3.]
 [  6.   4.  10.  13.   4.   5.   4.   8.   4.   9.]]
Aprovados
[[ 5.  2.  1. 38. 13. 12.  7.  7.  6.  6.]
 [ 4.  1. 10.  3.  6.  8.  4.  2.  3.  3.]
 [11.  9. 12.  8. 25.  5.  6.  6.  5.  5.]
 [12. 16.  2.  8.  0.  5.  2.  2.  3.  2.]
 [18.  2.  1.  6.  1.  9.  8. 11.  5.  6.]
 [ 5.  5.  4.  0.  3.  5.  2.  2. 11.  4.]
 [ 6.  3.  5.  5.  3.  4.  6.  6.  9.  4.]
 [ 2.  4.  6.  5.  7.  5.  7.  6. 18. 11.]
 [13.  6.  3.  5.  4.  5.  6.  6. 12.  3.]
 [ 6.  4.  7. 12.  4.  5.  3.  7.  4.  9.]]
```

Com as matrizes calculadas foi possível transformar as informações colhidas em gráficos como mostrado na figura:



Conclusão

Após a finalização do experimento foi possível identificar pelo menos dois grupos nos quais se observou uma probabilidade de reprovação em 100%, além de vários outros com probabilidades menores, porém significativas. Através desses resultados é possível ser tomada alguma ação com o intuito de diminuir os índices de reprovação na disciplina de Lógica de Programação.

Referências

Redes Neurais Auto-Organizáveis - SOM. Visto em http://aimotion.blogspot.com/2009/04/redes-neurais-auto-organizaveis-som.html Visitado em 15/10/2019.

sklearn.preprocessing.MinMaxScaler. Visto em: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html visitado em 16/10/2019

Minisom. visto em: https://github.com/JustGlowing/minisom. visitado em 16/10/2019.