Análise da situação de alunos matriculados na disciplina LoP utilizando Self Organized Maps

Introdução

Este trabalho foi desenvolvido por Igor Carvalho de Brito Batista, Rony de Sena Lourenço e Thatiana Jéssica da Silva Ribeiro.

O objetivo se trata de analisar a probabilidade de aprovação de alunos da turma de Lógica de Programação da Escola de Ciências e Tecnologia (ECT) da UFRN. Para realizar determinada análise, foi utilizado uma base de dados contendo informações dos semestres 2017.2, 2018.1, 2018.2 e 2019.1. Esta análise havia sido realizada anteriormente utilizando a rede MLP.

Metodologia

O modelo de aprendizado de máquina utilizado neste trabalho foi o Self Organized Maps (SOM).

Este é um método de aprendizado não supervisionado no qual o ferramental matemático é utilizado para promover o mapeamento de uma distribuição dos atributos de entrada de um espaço de alta dimensão para um de baixa dimensão.

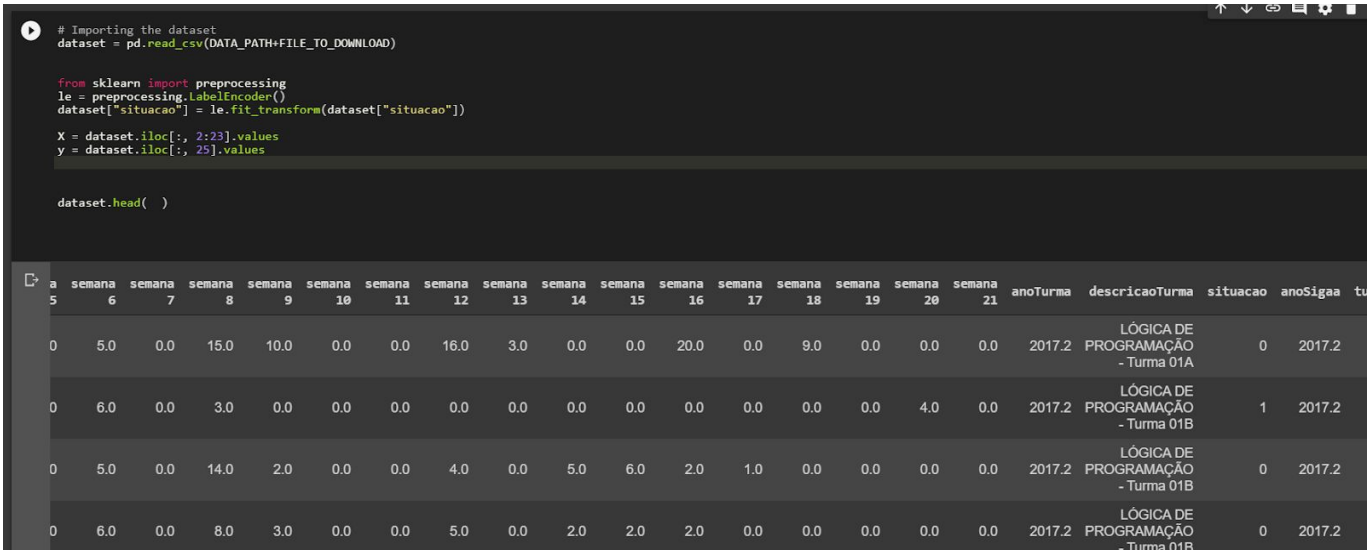
O funcionamento deste método acontece da seguinte maneira: os pontos do conjunto de dados de entrada competem entre si pela representação. O processo é iniciado com a inicialização dos vetores de peso. Então, um vetor de amostra é selecionado aleatoriamente e o mapa de vetores de peso é pesquisado para descobrir qual peso é o mais representativo para essa amostra. Esse peso é então denominado o “vencedor”.

Cada vetor de peso têm pesos vizinhos que estão próximos a ele. O peso escolhido é recompensado por se tornar mais parecido com o vetor de amostra selecionado aleatoriamente. Além disso, os vizinhos desse peso também são recompensados por se tornarem mais parecidos com o vetor de amostra escolhido. Assim, o mapa é continuamente atualizado.

Como o intuito é prever a probabilidade de aprovação, então buscamos selecionar os atributos que ajudassem o modelo a entender o comportamento de alunos exemplares. No caso deste estudo, foram escolhidas as submissões das atividades do alunos a cada semana.

Códigos

Para o desenvolvimento da atividade, foi utilizado o código cedido pelo professor em aula. O código faz uso das bibliotecas numpy, matplotlib, scikit-learn, keras e pandas. A biblioteca pandas foi utilizada para importar o dataset em csv contendo todos os dados referentes a quantidade de submissões e situação dos alunos. Em seguida, as colunas com a quantidade de submissões semanais foram selecionadas como os atributos.

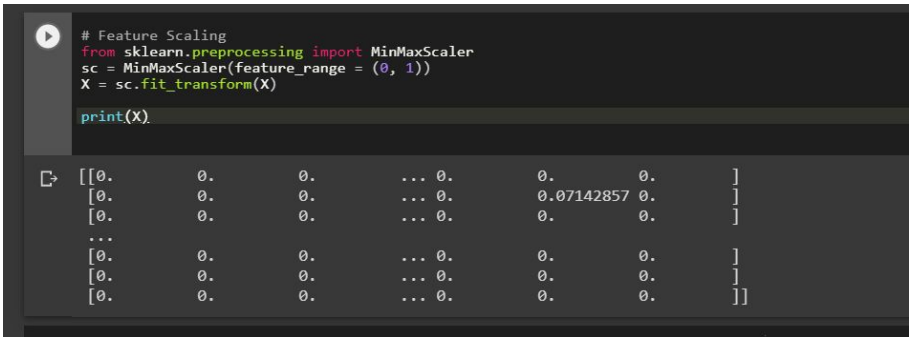


```
# Importing the dataset
dataset = pd.read_csv(DATA_PATH+FILE_TO_DOWNLOAD)


from sklearn import preprocessing
le = preprocessing.LabelEncoder()
dataset["situacao"] = le.fit_transform(dataset["situacao"])

X = dataset.iloc[:, 2:23].values
y = dataset.iloc[:, 25].values


dataset.head(  )
```

a	semana 5	semana 6	semana 7	semana 8	semana 9	semana 10	semana 11	semana 12	semana 13	semana 14	semana 15	semana 16	semana 17	semana 18	semana 19	semana 20	semana 21	anoTurma	descricaoTurma	situacao	anoSigaa	tu
0	5.0	0.0	15.0	10.0	0.0	0.0	16.0	3.0	0.0	0.0	20.0	0.0	9.0	0.0	0.0	0.0	2017.2	LÓGICA DE PROGRAMAÇÃO - Turma 01A	0	2017.2	
0	6.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	2017.2	LÓGICA DE PROGRAMAÇÃO - Turma 01B	1	2017.2	
0	5.0	0.0	14.0	2.0	0.0	0.0	4.0	0.0	5.0	6.0	2.0	1.0	0.0	0.0	0.0	0.0	2017.2	LÓGICA DE PROGRAMAÇÃO - Turma 01B	0	2017.2	
0	6.0	0.0	8.0	3.0	0.0	0.0	5.0	0.0	2.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0	2017.2	LÓGICA DE PROGRAMAÇÃO - Turma 01B	0	2017.2	

Figura 01 - Escolha das colunas da tabela.

É interessante notar a partir da Figura 01 que os rótulos da situação do aluno foi modificado de strings para valores inteiros, no qual : 0 representa alunos aprovados, 1 aprovados por nota, 2 reprovado, 3 reprovado por nota e 4 reprovado por média e falta. Os atributos são então normalizados, como mostra a Figura 2.

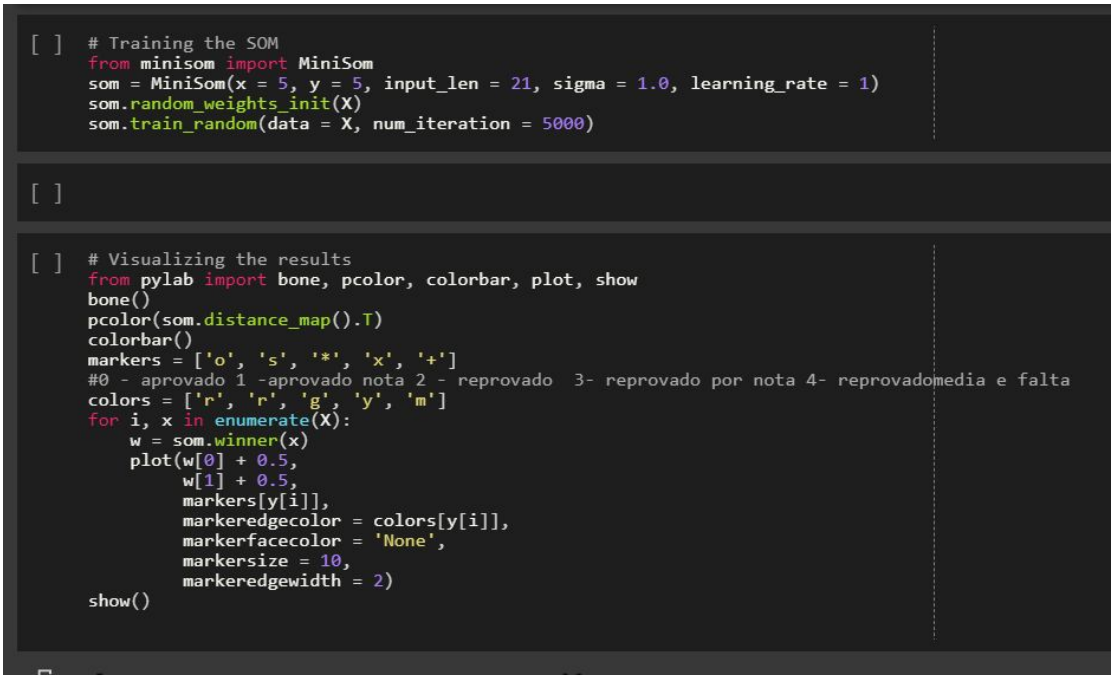


```
# Feature Scaling
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
X = sc.fit_transform(X)

print(X)
```

```
[[0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.07142857 0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 ...
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]]
```

Figura 02 - Normalização

Na figura a seguir é possível observar a seleção dos parâmetros da rede neural. Inicialmente foi utilizado uma malha de 5x5, com 21 atributos de entrada, sigma e taxa de aprendizado igual a unidade.



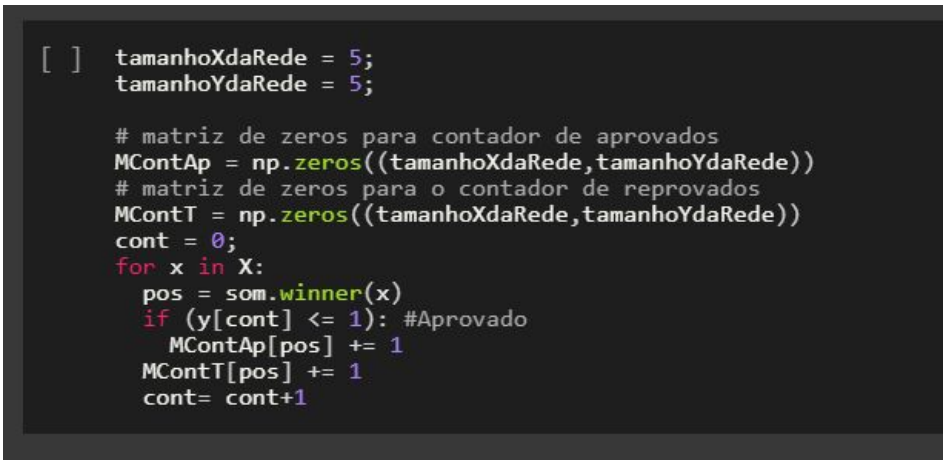
```
[ ]  # Training the SOM
     from minisom import MiniSom
     som = MiniSom(x = 5, y = 5, input_len = 21, sigma = 1.0, learning_rate = 1)
     som.random_weights_init(X)
     som.train_random(data = X, num_iteration = 5000)
```

```
[ ]
```

```
[ ]  # Visualizing the results
     from pylab import bone, pcolor, colorbar, plot, show
     bone()
     pcolor(som.distance_map().T)
     colorbar()
     markers = ['o', 's', '*', 'x', '+']
     #0 - aprovado 1 -aprovado nota 2 - reprovado  3- reprovado por nota 4- reprovadomedia e falta
     colors = ['r', 'r', 'g', 'y', 'm']
     for i, x in enumerate(X):
         w = som.winner(x)
         plot(w[0] + 0.5,
              w[1] + 0.5,
              markers[y[i]],
              markeredgecolor = colors[y[i]],
              markerfacecolor = 'None',
              markersize = 10,
              markeredgewidth = 2)
     show()
```

Figura 3 - Treinamento da rede SOM.

Foi notado que a visualização padrão deste método era um pouco confusa, então o seguinte trecho de código mostrado na Figura 4 foi adicionado, a fim de a partir da criação de uma matriz de aprovação obter-se melhores visualização em gráficos do tipo pizza e barras verticais.



```
[ ]  tamanhoXdaRede = 5;
     tamanhoYdaRede = 5;

     # matriz de zeros para contador de aprovados
     MContAp = np.zeros((tamanhoXdaRede,tamanhoYdaRede))
     # matriz de zeros para o contador de reprovados
     MContT = np.zeros((tamanhoXdaRede,tamanhoYdaRede))
     cont = 0;
     for x in X:
       pos = som.winner(x)
       if (y[cont] <= 1): #Aprovado
         MContAp[pos] += 1
       MContT[pos] += 1
       cont= cont+1
```

Figura 4 - Criação da matriz de aprovados.

A Figura 5 mostra a matriz obtida e a criação da visualização em gráfico pizza.

```
[ ]  print("Total:")
     print(MContT)


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

```
Total:
[[  8.  34.  22.  19.  34.]
 [  8.  26.  19.  13.  13.]
 [ 23.  33.  32.  34.  52.]
 [ 26.  32.  33.  57.  71.]
 [ 46.  33.  28. 200.  52.]]
Aprovados
[[ 8. 32. 21. 15. 33.]
 [ 8. 23. 14. 12. 13.]
 [22. 28. 31. 29. 40.]
 [24. 27. 24. 34. 29.]
 [43. 25. 19. 52. 33.]]
```

```
[ ]
     cont = 1;
     for i in range(len(MContT)):
       for j in range(len(MContT)):
         plt.subplot(tamanhoXdaRede,tamanhoYdaRede,cont)
         cont=cont+1
         sizes = [MContAp[i][j],MContT[i][j]-MContAp[i][j]]
         plt.pie(sizes)
     plt.show()
```

Figura 5 - Matriz de aprovados.

Para analisar somente um único neurônio, a seguinte função, mostradas na Figura 6, foi criada.

```
[ ]  quantidadeCaracteristicas = 21
     pesos = som.get_weights()
     def mostraNeuronio(linha,coluna):
       # Mostra um peso
       x = np.arange(quantidadeCaracteristicas)
       plt.bar(x, pesos[linha,coluna,:])
       plt.axis([-1, 21, 0, 1])
       plt.show()
```

Figura 6 - Função para análise de um neurônio.

Resultados

Utilizando o código descrito na seção anterior, o seguinte mapa dos clusters, mostrado na Figura 7, foi obtido.

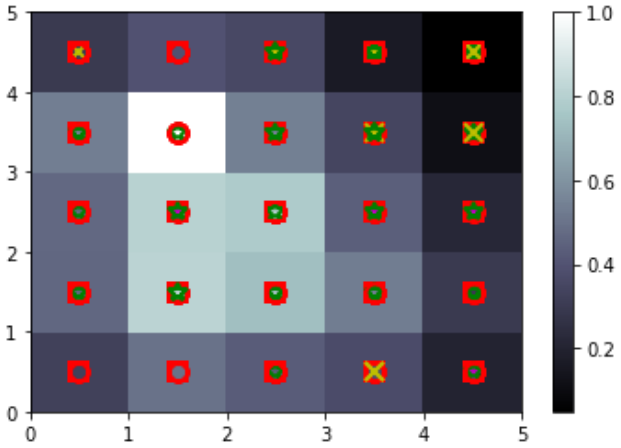


Figura 7 - Mapa com clusters distintos.

Em geral, é esperado que se a distância média entre os neurônios for alta, os pesos ao redor serão muito diferentes e uma cor clara é atribuída à localização do peso. Se a distância média for baixa, é atribuída uma cor mais escura (amostras pertencentes então ao mesmo cluster). A coloração mais clara indicaria então a divisão entre os clusters. Entretanto, como não ficou tão claro a divisão entre eles, podendo ser percebido isso até pela confusão nos marcadores relativos à situação do aluno, foi escolhido gerar a visualização em gráfico pizza mostrada na Figura 8.

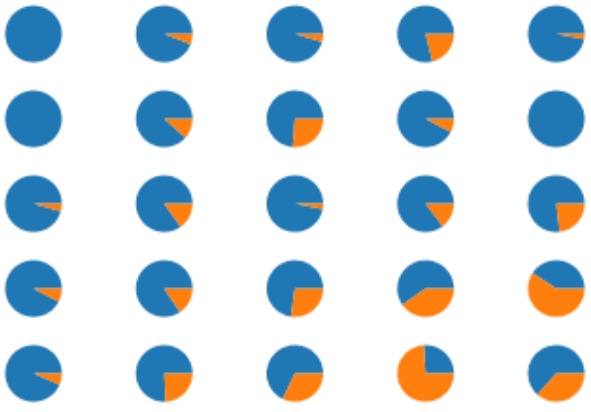


Figura 8 - Resultados em representação de gráfico pizza.

A coloração azul na Figura 08 está relacionada com alunos aprovados, enquanto a coloração alaranjada a alunos reprovados. Analisando isoladamente o neurônio representado na matriz nos índices (0,0) podemos perceber que os alunos com 100% de aprovação apresentam o seguinte perfil de submissões, mostrado na Figura 9.

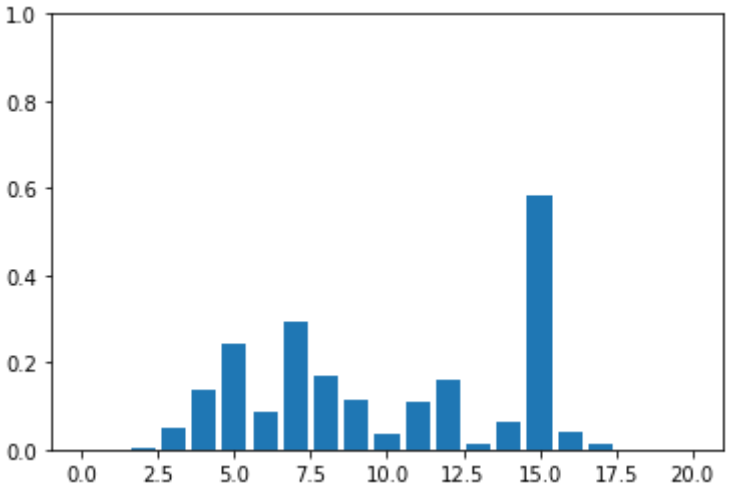


Figura 09 - Resultados em representação de gráfico em barras verticais.

A partir Figura 09 é possível observar que alunos que apresentam boa sequência de submissões tendem a ter alta percentagem de aprovação. Já para um aluno com 75 % de reprovação, mostrado nos índices (4, 3), é possível notar o seguinte perfil de submissões, mostrado na Figura 10.

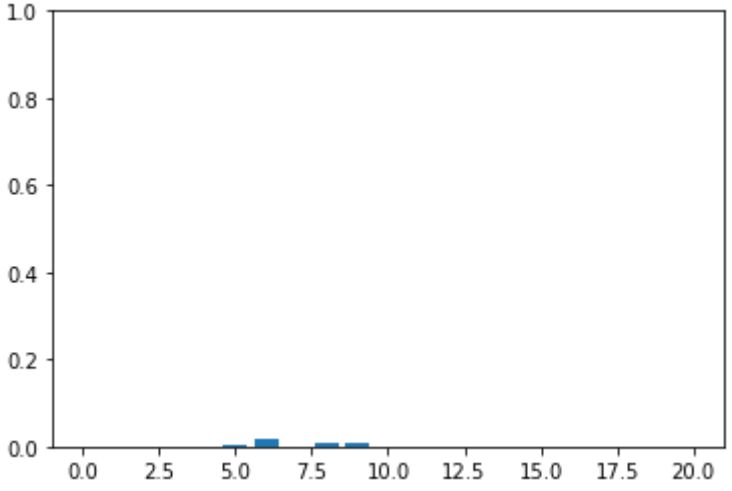


Figura 10 - Resultados em representação de gráfico em barras verticais.

É possível inferir a partir da Figura 10 que quando a frequência de submissão das atividades diminui, a probabilidade do aluno reprovar aumenta. Na Figura 11 é mostrado o gráfico de frequência de submissões de atividade para todos os neurônios.

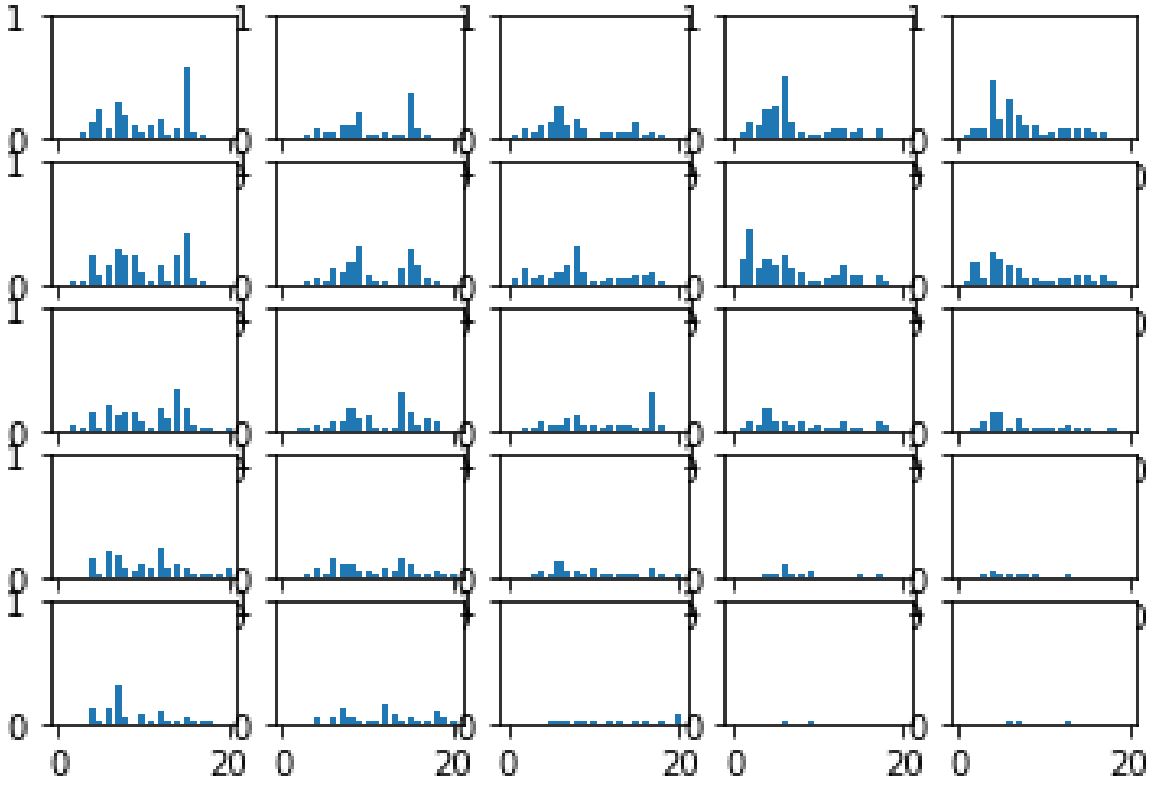


Figura 11 - Resultados em representação de gráfico em barras verticais.

Conclusão

Com base nos resultados mostrados é possível concluir que o agrupamento das amostras condiz com a frequência de envios, alunos com menores submissões, tendem a estar associados à situação de reprovação. Além disso, a SOM é uma rede neural bastante interessante, visto que ela utiliza um aprendizado baseado em competição, ao invés de um aprendizado baseado na correção de erros, como é observado em outras redes neurais.