Previsão de perfil de alunos com base em questões submetidas

Introdução

O grupo é composto por Tobias Aguiar e Leandro Rodrigues, onde o relatório em questão é feito com o objetivo de analisar e compreender o conteúdo estudado em sala fazendo experimentos e previsões com modelos de *Machine Learning* e analisando seus respectivos resultados. O problema a ser estudado é : Utilizando um modelo de aprendizagem não supervisionada, como conseguir traçar os diferentes perfis de estudantes que existem baseado nas questões em que eles submetem? Para auxiliar na busca da solução, foi utilizada uma base de dados onde esta contém o número de questões submetidas em diferentes semanas durante o semestre como também a situação final de cada aluno.

Metodologia

Para o relatório em questão, utiliza-se o modelo da rede SOM (*Self-Organizing Maps*), proposto por Kohen, um modelo de aprendizado de máquina não supervisionado. Esta utiliza um modelo matemático para visualização de dados que possui alta dimensionalidade, realizando uma transformação de espaço, mapeando uma distribuição de alta-dimensão em uma grade regular de baixa dimensão, sem deixar de preservar métricas e topologias dos dados originais. Rede bastante utilizada em aplicações como reconhecimento de voz, visualização de registros financeiros, dentre outros.

O algoritmo para a construção consiste nos seguintes passos : inicializar a rede com parâmetros aleatórios de valor entre o mínimo e máximo das característica dos dados, em seguida deve-se escolher um dado qualquer e encontrar o neurônio “vencedor”, isto é, aquele que cuja distância entre o conjunto de características dos dados e dos pesos do neurônio é a menor. Uma vez feito, atualiza-se os pesos dos neurônios vizinhos àquele eleito “vencedor”, e, por fim, repete-se o primeiro passo até que o critério de parada desejado seja atingido.

Em relação às etapas de treinamento e teste, deve-se importar, para auxílio, a biblioteca que contém os métodos relacionados a rede SOM, denominada Minisom. Deve-se selecionar o tamanho da rede para formar uma matriz, o tamanho

da entrada, relacionado à quantidade de características existentes. Depois de obter a rede treinada, deve-se construir a matriz de pesos de cada neurônio, para depois poder estimar o neurônio “vencedor” e atualizar seus vizinhos mais próximos.

Os atributos selecionados, em resumo, foram as questões submetidas até, no máximo, a quinta semana de aula, pois deseja-se traçar perfis de alunos o mais antes possível, logo convém selecionar questões do início do semestre. // (... melhorar)

Códigos

O treinamento do método som é feito com alguns passos, conforme a figura 01. O primeiro passo feito foi classificar a base de dados, em entrada e saída, no caso a entrada é a submissões e a saída é a situação de aprovado ou reprovado. Depois é definido o tamanho da rede e a quantidade de características para que sejam entradas da função “mini som”, em que os pesos irão se ajustando até o número máximo de interação.

Figura 01- Código referente a definições iniciais do método som.

```
[ ]
    X_train = dataSet.iloc[:, 2:22].values
    target_train = dataSet.iloc[:,25].values

    [row, col] = X_train.shape
    print (row," ",col)

    print(X_train[1,:])
```

```
948   20
[0. 0. 0. 5. 0. 6. 0. 3. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 4.]
```

```
[ ]  # Training the SOM
     tamanhoXdaRede = 6;
     tamanhoYdaRede = 6;



     quantidadeCaracteristicas = col
     from minisom import MiniSom
     som = MiniSom(x = tamanhoXdaRede, y = tamanhoYdaRede, input_len = quantidadeCaracteristicas, sigma = 1.0, learning_rate = 0.4)
     som.pca_weights_init(X_train)
```

```
[ ]  som.train_random(data = X_train, num_iteration = 40000)
```

```
[ ]  import matplotlib.pyplot as plt
     import numpy as np
     # Obtem o vetor de pesos da rede treinada
     pesos = som.get_weights()
```

O intuito do segundo trecho de código é definir a matriz de total de submissões e aprovados(figura 02). Como é visto, é obtido a matriz vencedora, aquela que melhor representa a entrada; a partir dela, em cada neurônio é feito uma contagem de quantos foram aprovados e quantos submeteram, e então, cria as duas matrizes com as contagens. Isso possibilita relacionar, a partir da matriz representativa gerada pelo método som, o quanto que o número de submissões resulta em aprovação.

Figura 02- Trecho do código que define as matrizes de aprovação e submissões.

```
# encontra o vencedor
x = X_train[1,:]
pos = som.winner(x)


# 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_train:
  pos = som.winner(x)
  if (Y_train[cont] <= 1): #Aprovado
    MContAp[pos] += 1
  MContT[pos] += 1
  cont= cont+1
```

Experimentos

Os testes executados foram em relação a matriz som, no qual gerou uma matriz representativa dos dados. Na figura 3 é visto a quantidade de aprovados e reprovados em cada neurônio da matriz som, azul é aprovado e laranja reprovado. A figura 4 mostra a quantidade de submissão. Por exemplo, no neurônio da coluna 2 e linha 2, comprova que pouca submissão resulta em reprovação; por outro lado, na coluna 6 e linha 3 todos foram aprovados e houve muitas submissões.

O método som possibilita criar classes, que nesse caso foram duas. Assim, com esses resultados, cria-se um caminho muito provável para a aprovação: submeter muitos exercícios. Em poucos casos, há exemplos de poucas submissões e aprovado, isso pode se dar pelo fato de que muitos alunos já têm conhecimento prévio em lógica de programação.

Figura 03-Reprovados e aprovados Figura 4- Total de submissão

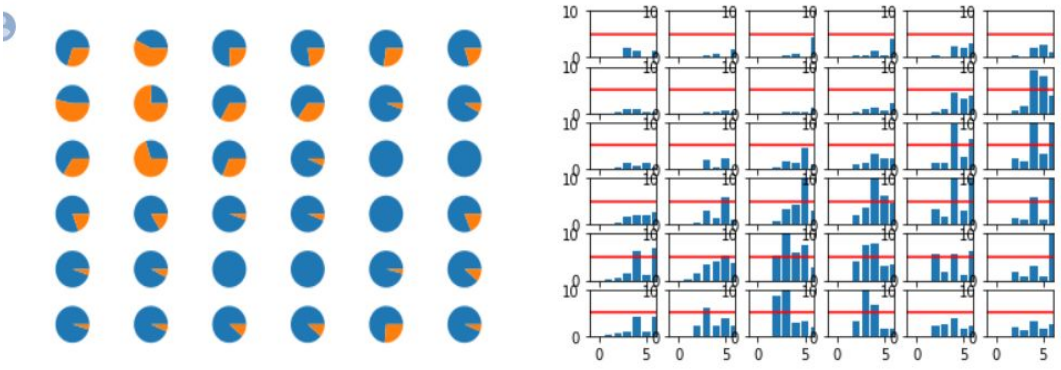

Figura 04- Relação entre aprovados e total de submissões.