UNIVERSIDADE FEDERAL DO ESTADO DO RIO GRANDE DO NORTE

Disciplina / Área de conhecimento:

ECT2702 - TÓPICOS AVANÇADOS EM INFORMÁTICA I - T01 (2019.2)

MLP - Multi-Layer Perceptron

Luís Fernando Tavares

Natal, 2019

**Sumário**

1 Introdução…………………………………………...…………...…………..............................………......03

## 2 Metodologia.……………...……………………….......................................................….……......04

## 3 Códigos……………………....................................................................................………......05

## 4 Experimentos………………..........................................................…....................………......09

02

**1 Introdução**

O problema abordado neste relatório é: uma base de dados que contem registro de alunos da disciplina de LOP (Lógica de Programação), contendo informações sobre as atividades e provas feitas por eles em um semestre, o objetivo é saber se o aluno, de acordo com os dados obtidos até a metade do semestre, ira ser aprovado na disciplina.

A base de dados possui diversos dados coletados dos alunos durante o semestre, tais como:

* qsemana: Quantidade de dias diferentes que ocorreram.
* submeteu: quantidade de questões submetidas pelo aluno por semana.
* subListaLab: quantidade de submissões na lista de laboratórios a cada 2 semanas.
* subListaExer: quantidade de submissões na lista de exercícios a cada 2 semanas.
* subDistintasLab: quantidade de dias em que o submeteu a cada 2 semanas nas listas de laboratório.
* subDistintasExer: quantidade de dias em que o envio será enviado a cada 2 semanas nas listas de exercícios.
* diferentesLabSemanas: quantidade de questões diferentes submetidas nas listas de laboratório a cada 2 semanas.
* diferentesExerSemanas: quantidade de questões diferentes submetidas nas listas de laboratório a cada 2 semanas.
* situação: indica se o aluno foi aprovado (1) ou reprovado (0).
* qsub: quantidade de submissões das listas correspondentes (seja laboratório ou exercício).
* qsubp: quantidade de questões submetidas para a prova 1 (L1, L2, L3) ou a prova 2 (L4, L5).
* totalsub: total de submissões feitas pelo aluno.
* igualassem: quantidade de submissões em que o aluno acertou 100%, seja nas L123 ou L45.

03

**2 Metodologia**

O modelo de Machine Learning utilizado para resolver este problema é o Perceptron Multicamadas (PMC ou MLP — Multi Layer Perceptron), ele é uma rede neural com uma ou mais camadas ocultas com um número indeterminado de neurônios. A camada oculta possui esse nome porque não é possível prever a saída desejada nas camadas intermediárias.

Diferentemente do Perceptron, onde existe apenas um único neurônio de saída, a MLP pode relacionar o conhecimento a vários neurônios de saída.

No Perceptron apenas multiplicamos os pesos e adicionamos o viés, mas fazemos isso em apenas uma camada.

Atualizamos o peso quando encontramos um erro na classificação ou não classificado. Equação de atualização de peso é esta:

peso = peso + taxa de aprendizado \* (esperado - previsto) \* x

Ele funciona a partir de um estimulo que é propagado da entrada para saída passando por cada camada escondida.

As etapas de treinamento são divididas em três partes:

**1º** Forward pass (Passe avançado):

Nesta etapa do treinamento do modelo, apenas passamos a entrada (atributo) para o modelo e multiplicamos com pesos e adicionamos viés em cada camada e encontramos a saída calculada do modelo.

**2º** Calculate error or loss (Calcular erro ou perda):

Quando passamos a instância de dados (ou um exemplo), obtemos alguma saída do modelo chamado saída prevista e temos o rótulo com os dados que são saída real ou saída esperada. Com base nesses dois, calculamos a perda que precisamos retropropagar (usando o algoritmo Backpropagation). Existem várias funções de perda que usamos com base em nossa produção e exigência.

**3º** Backward pass (Passe para trás):

Após calcular a perda, retropropagamos a perda e atualizamos os pesos do modelo usando gradiente. Nesta etapa, os pesos serão ajustados de acordo com o fluxo do gradiente nessa direção.

Os atributos escolhidos para o MLP são a nota da primeira prova, quantidade de questões submetidas para a prova 1 (Lista 1, Lista 2, Lista 3) e quantidade de submissões em que o aluno acertou 100% nas Listas 1, 2 e 3, foram escolhidos pois são os que possuem maior impacto no resultado da nota e preparação da prova na primeira metade do semestre.

04

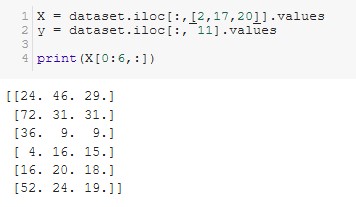
**3 Códigos**

**1º Importação:**



Esse código faz a importação dos dados da base de dados, no caso em .csv, denominada de DataBaseLop. O dataset.head mostra uma certa quantidade de dados contidos em cada linha da base.

**2º Seleção dos atributos:**

****

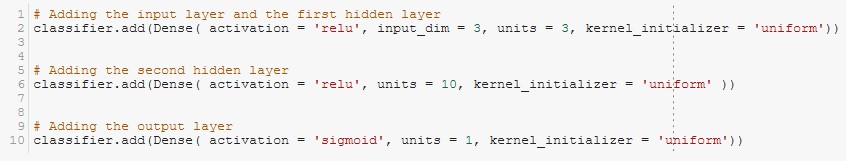
Esse código permite a escolha dos atributos (X) que são usados para comparação do parâmetro (Y), sendo este a situação dos alunos, se estão aprovados ou não. O print mostra uma certa quantidade de dados dos atributos contidos em cada linha.

**3º Treinamento e teste:**

****

Esse código inicia a separação dos atributos para serem treinados e testados. O test\_size é a porcentagem de atributos que irão ser utilizados para o teste (No caso é 20%).

**4º Camadas:**

****

05

Esse código cria as camadas (Neurônios) para serem processados os atributos, a camada inicial possui a função de ativação da camada (Rectifier\*), a entrada de certo número de atributos, quantidade de neurônios presentes e a função que deve ser inicializada com os pesos e atualiza-los (Função Uniforme).

A segunda camada tem a função de ativação da camada (Rectifier\*), quantidade de neurônios presentes e a função que deve ser inicializada com os pesos e atualiza-los (Função Uniforme).

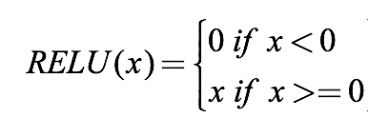
A terceira e última camada tem a função de ativação da camada (Sigmoid\*), quantidade de neurônios de saida e a função que deve ser inicializada com os pesos e atualiza-los (Função Uniforme).

5**º Treinamento:**

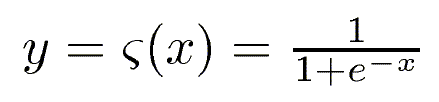
****

Esse código inicia o treinamento na MLP que permite a estimativa do gradiente, o batch\_size é um hiperparâmetro de descida de gradiente que controla o número de amostras de treinamento a serem trabalhadas antes que os parâmetros internos do modelo sejam atualizados, quanto menor o batch\_size, menor é acurácia do gradiente, porem utilizara menos memoria e será mais rápido, e vai ditar quantas interações levara para completar um epoch é um hiperparâmetro de descida de gradiente que controla o número de passagens completas pelo conjunto de dados de treinamento.

\*

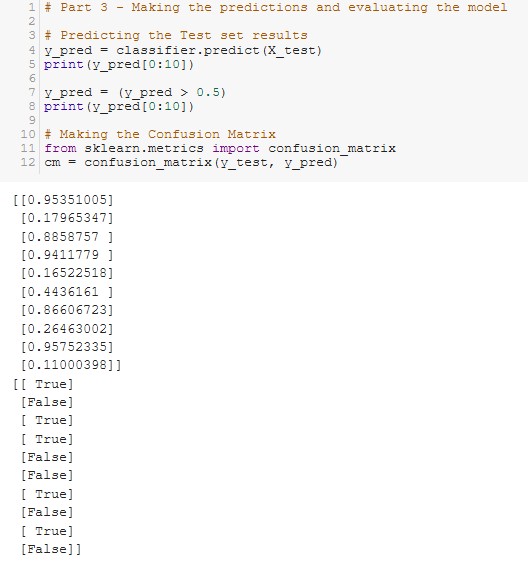
Onde x é a entrada para um neurônio. Isso também é conhecido como função de rampa.

\*

 Onde x é a entrada para um neurônio.

06

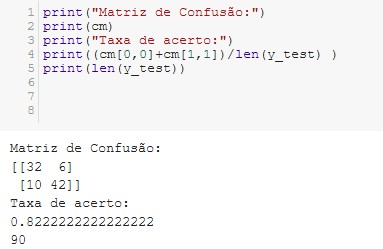
6**º Previsões e Matriz de confusão:**

****

Esse código mostra a previsão dos dados feitos pelos testes em comparação com o trainamento e a construção da matriz de confusão.

07

7**º Taxa de acerto:**

****

Calcula e mostra a taxa de acerto e os valores da matriz de confusão (verdadeiro positivo, falso negativo, falso positivo e verdadeiro negativo).

08

**4 Experimentos**

Os tipos de testes executados foram para, de acordo com os dados fornecidos, ter um gral de certeza de se o aluno será reprovado ou não na disciplina, utilizando duas camadas escondidas e uma de saída, as duas primeiras com a função de ativação chamada Rectifier e a última com a sigmoide, a porcentagem de atributos que irão ser utilizados para o teste é de 20% (test\_size), e para descobrir a estimativa do gradiente, o batch\_size = 5 e os epochs = 80, por fim os testes são comparados com o trainamento e é construida a construção da matriz de confusão que mostra quantidades de verdadeiro positivo, falso negativo, falso positivo e verdadeiro negativo, que são respectivamente quanto dos dados bateram, bateram mas não foram considerados, não bateram mas foram considerados e os que não bateram.

Parâmetro avaliado foi a situação de aprovado e não aprovados na diciplina em um determinado semestre, o qual foi comparado com os seguintes atributos: a nota da primeira prova, quantidade de questões submetidas para a prova 1 (Lista 1, Lista 2, Lista 3) e quantidade de submissões em que o aluno acertou 100% nas Listas 1, 2 e 3, pois são os que mais pesam na previsão do resultado final com os dados da primeira metade do semestre.

Os resultados obtidos foram a matriz de confusão, a qual demonstrava os acertos, erros e falhas obtidas através da comparação entre as taxas dos testes e do que foi previsto, a qual nos permitiu calcular a taxa de acerto, a qual mostrava o quanto o programa era certeiro para calcular os alunos que iriam ser reprovados ou não, no caso possui uma taxa de acerto de 83.3% , ou seja foi possível prever a situação de 83% dos alunos de um determinado semestre, assim esse programa pode ser usado para prever a situação de uma turma e alerta-los para que melhorem sua situação.

09