UFV - Universidade Federal de Viçosa
DPI - Departamento de Informática
Prof. Alcione

INF 110 - Programação I
Trabalho 2
Prazo: 16/11/2020

O trabalho consiste em projetar e implementar programas em C ou C++ para resolver os 2 problemas descritos adiante. Não há necessidade de entregar os códigos impressos ou por e-mail, nem entregar algum relatório. Os códigos devem ser submetidos para correção no sistema Boca, e para cada problema podem ser feitas tantas submissões quantas forem necessárias até obter uma correta. No caso de mais de uma submissão correta para um problema, será considerada na avaliação principalmente a última.

ATENÇÃO!

Este trabalho é **individual**.

Duas importantes etapas na resolução de cada problema são o planejamento da lógica do algoritmo e a implementação dessa lógica em uma linguagem de programação. Antes de buscar alguma ajuda na resolução do trabalho, considere os seguintes pontos:

1. Se alguém te contar a lógica que desenvolveu antes de você tentar desenvolver uma, perderá a oportunidade de exercitar esta etapa.

2. Se alguém te mostrar ou passar uma solução em código C ou C++ antes de você tentar desenvolver ou encontrar os erros na sua implementação, perderá a oportunidade de exercitar e desenvolver suas habilidades de programação.

Portanto, tente desenvolver por si mesmo a lógica do algoritmo de solução. Se não conseguir resolver uma questão completamente, busque ajuda do professor ou de algum colega, mas não olhe um código pronto.

E quando terminar um código jamais passe esse código para um colega. Ajude-o explicando a lógica da solução, ou encontrando erros na implementação, jamais entregue seu código a outros alunos. Trabalhos com códigos idênticos ou muito parecidos receberão nota **ZERO**.

Comentários e indentação

Códigos que não estejam devidamente indentados podem receber desconto na nota. Além disso, devem existir comentários explicativos nas partes mais complexas da solução.

Data limite

A data limite de entrega será dia 16 de novembro. Ainda falta bastante tempo, mas comece a fazê-lo assim que receber este enunciado para evitar problemas de última hora.

Link para o envio dos programas no sistema Boca

http://boca.dpi.ufv.br/trabalhos
Name: seu número de matrícula
Password: seu número de matrícula
ATENÇÃO! **Troque sua password!** (link Options no menu do Boca)

Problema A. Jogo da velha

Arquivo-fonte: `velha.c` ou `velha.cpp`

Desenvolva um programa que receba uma sequência de jogadas do jogo da velha e emita uma mensagem indicando o ganhador ou se houve empate (mensagem "Velha!"). Assuma que a sequência de jogadas é um jogo completo, ou seja, existe um vencedor ou é empate. Use uma matriz para representar o jogo da velha.

Entrada

A primeira linha da entrada contém um valor inteiro P, que é o número de jogadas a serem lidas. Em seguida cada linha será uma jogada no formato `jogador linha coluna`, onde `jogador` será o caracter 'x' ou 'o', `linha` será um valor de 0 a 3, representando a linha da jogada, e `coluna` será um valor de 0 a 3, representando a coluna da jogada.

Saída

Seu programa emitirá uma única linha com uma quebra de linha no final indicando o vencedor.

Exemplos

Entrada	Saída
5 x 0 0 o 1 0 x 0 1 o 1 1 x 0 2	Venceu x!

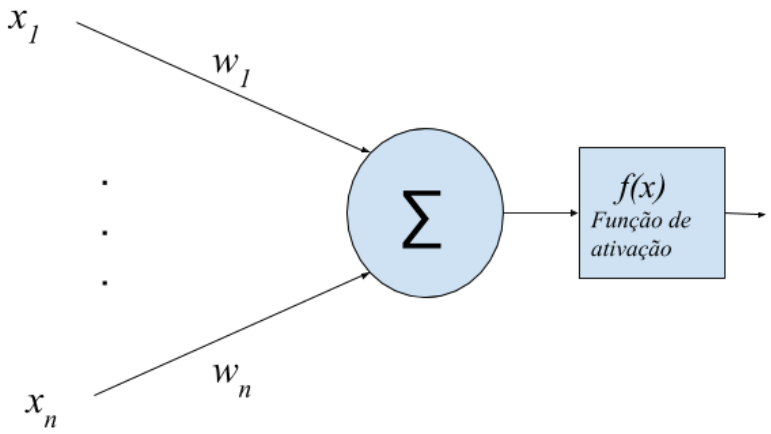
Entrada	Saída
6 x 0 1 o 0 0 x 0 2 o 1 1 x 2 0 o 2 2	Venceu o!

Entrada	Saída
9 x 0 1 o 0 0 x 0 2 o 1 1 x 2 2 o 2 0 x 1 0 o 1 2 x 2 1	Velha!

Problema B. Neurônio

Arquivo-fonte: `neuronio.c` ou `neuronio.cpp`

Um neurônio artificial é um classificador que é capaz de aprender a classificar a partir de exemplos. O formato geral de um neurônio artificial é o seguinte:



Onde x_i representa a entrada e w_i representam os pesos.A função somatória opera de acordo com a fórmula que calcula produto interno entre os vetores de entrada e pesos, expresso pela fórmula:

$$\mathbf{x}.\mathbf{w} = \sum_{i=1}^{N} x_i w_i = x_1 \times w_1 + x_2 \times w_2 + \cdots x_N \times w_N = z$$

A **função de ativação** é uma função que é aplicada ao valor de saída do produto interno (z), para emitir um valor entre 0 e 1. Uma função muito utilizada é a função **sigmoide**:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

O aprendizado do neurônio é, basicamente, o ajustes dos pesos para que o neurônio emita o resultado desejado para os valores de entrada. O ajuste no peso é feito em função da diferença entre o valor esperado e o previsto para cadas exemplo. Ou seja, é feito em função do erro da previsão. Por exemplo, se para uma entrada a saída esperada era 1 e a rede emitiu 0.8, então o erro é de 0.2 e o ajuste dos pesos é feito para minimizar esse erro. Esse ajuste, chamado de **treinamento**, é feito gradualmente, sendo feito repetidas vezes para cada exemplo. Uma iteração completa por todos os exemplos de treinamento é chamada de **época** é são necessárias muitas épocas para treinar um neurônio. O ajuste do peso so neurônio é feito segundo a seguinte fórmula:

$$w_i = w_i + \alpha \times erro \times \sigma(z)(1 - \sigma(z))x_1$$

Onde α é uma taxa de aprendizado (usaremos 0.1), $erro$ é $y - \sigma(z)$, e y é o valor esperado. A expressão $\sigma(z)(1 - \sigma(z))$ é a derivada (inclinação) da função sigmoide e indica a direção do ajuste a ser realizado.

Depois de treinado o neurônio pode receber valores novos para realizar previsões sobre valores que não constam no conjunto de treinamento.

O pseudocódigo para treinamento e execução do neurônio seria o seguinte:

```
para cada época faça
     para cada exemplo faça
```

$$z = x_1 \times w_1 + x_2 \times w_2 + \cdots x_N \times w_N$$

$$\sigma(z)$$

$$erro = y - \sigma(z)$$

$$w_i = w_i + \alpha \times erro \times \sigma(z)(1 - \sigma(z))x_1$$

```
     fim para
fim para

// Testando novas entradas
Leia os valores de entrada
Calcule a saida aplicando
imprima σ(x1 × w1 + x2 × w2 + ··· xN × wN)
```

Desenvolva um programa que implemente o pseudocódico acima de modo que possa prever se um empréstimo deve ser concedido ou não. O neurônio possui duas entradas: o salário é numeros de salários mínimos e o valor do emprestimo em número de salários mínimos. São 10 entradas para treinamento, que devem ser codificadas internamente no programa, ou seja, não seriam lidas. Os vetores de treinamento seriam:

```
double x[2][n] = {{2.7 ,1.5 ,5.5 ,3.5 ,3.1 ,7.6 ,1.5,6.9,8.6,7.66 },
                  {10.5,11.8,20.0,15.2,14.5,14.5,3.5,8.5,2.0,3.5 }};
double y[n] ={0   ,0   ,0   ,0   ,0   ,1   ,1  ,1  ,1  , 1   }
```

Onde, `x[0][n]` são os valores do salário, `x[1][n]` são os valores do pedido do emprestimo e `y` são os valores de saída, sendo 0 para empréstimo negado e 1 para empréstimo concedido.

Observações

A saída do neurônio deve ser alterada para emitir somente 0 ou 1. Sendo assim o programa deve emitir 1 se $\sigma(z) > 0.5$ e 0 caso contrário.

Use 0.1 como valor de α, 0.5 como valor inicial para w_1 e w_2, e use 1000 para o número de épocas.

Entrada

A entrada será simplesmente dois valores reais representando, respectivamente o salário e o pedido de empréstimo em termos de salário mínimo.

Saída

Seu programa deve gerar uma linha na saída contendo 0 ou 1 seguido de uma quebra de linha.

Exemplos

Entrada	Saída
2.0 8.0	0

Entrada	Saída
3.0 6.0	1

Entrada	Saída
8.5 3.8	1

Entrada	Saída
5.0 20.0	0