UNIVERSIDADE FEDERAL DE ITAJUBÁ

CIÊNCIA DA COMPUTAÇÃO

SIMULAÇÃO DE UM NEURÔNIO ARTIFICIAL

2023002135 - Caio Mendes Ribeiro da Rosa
2023001272 - Davi Dias Monsores dos Santos
2022012988 - Isaac Davi Mendonça Viana
2023013578 - Raphael Rodrigues Pereira
2023009225 - Tiago de Figueiredo Reis

CAHC04 - PROJETO INTEGRADO (SEMINÁRIO)

Prof. Rafael Frinhani

INSTITUTO DE MATEMÁTICA E COMPUTAÇÃO
UNIFEI - Itajubá

2023

UNIVERSIDADE FEDERAL DE ITAJUBÁ
INSTITUTO DE MATEMÁTICA E COMPUTAÇÃO
CIÊNCIA DA COMPUTAÇÃO
CAHC04 - PROJETO INTEGRADO (SEMINÁRIO)

Simulação de um Neurônio Artificial

1 Introdução

A simulação de um neurônio artificial é um campo de estudo que busca reproduzir o comportamento dos neurônios biológicos em sistemas computacionais.

Neste artigo, será explorado o modelo pioneiro de um neurônio artificial, o modelo de McCulloch e Pitts e o Perceptron de Rosenblatt, bem como suas relações com o neurônio biológico. Além disso, serão abordados as principais aplicações de um conjunto de neurônios artificiais, ou seja, de uma rede neural artificial (RNA). Também será apresentado um exemplo de algoritmo para implementação prática. Nesse viés, também será discutido a relação entre a simulação de neurônios artificiais e redes neurais com as disciplinas do 1º período curso de Ciência da Computação.

2 Referencial Teórico

2.1 O Neurônio Biológico

O cérebro humano possui aproximadamente 10 bilhões de neurônios, células básicas do sistema nervoso, com a função de realizar a propagação do impulso nervoso, sendo responsável por controlar todas as funções e movimentos do organismo.

Os neurônios são interconectados através de sinapses, as quais permitem a passagem do impulso nervoso de um nerônio para outro através da liberação e do reconhecimento de substâncias químicas conhecidas como neurotransmissores como o Sódio (Na^+) e Potássio (K^+).

Ao receber um impulso, o neurônio o processa no corpo celular e o transmite pelo axônio, isso ocorre devido a um processo de despolarização que ocorre ao longo da membrana do neurônio. Os neurônios apresentam três componentes básicos:

- Dendritos: Recebem estímulos transmitidos por outros neurônios;
- Corpo celular (Soma): Coleta e combina informações vindas de outros neurônios e passam pelos dendritos, é basicamente a unidade de processamento de dados do neurônio;
- Axônio: Responsável pela transmissão de impulsos nervosos a outros neurônios. Klein & Martins (2006)

2.2 O modelo de McCulloch e Pitts

Esse modelo é baseado em uma simplificação do funcionamento do neurônio biológico. O modelo apresenta n entradas (x_1,x_2,...,x_n), pesos acoplados a cada entrada (w_1,w_2,...,w_n), uma

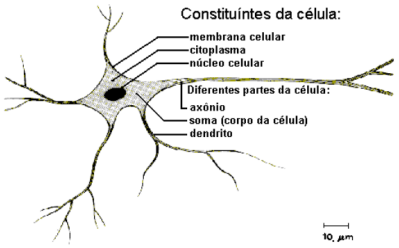

Figura 1: Representação de um neurônio . Fonte: Klein & Martins (2006).

unidade central de processamento e uma saída (y). O processamento (net) consiste em cada sinal de entrada apresentado multiplicado pelo valor do peso correspondente. Em seguida, é feito um somatório dos sinais multiplicados por seus respectivos pesos.

$$net = w_1x_1 + w_2x_2 + ... + w_nx_n = \sum_{j=1}^{n} w_jx_j$$

Se o valor dessa soma ultrapassar um limiar μ, o valor na saída é y=1, se não ultrapassar o valor da saída, é y=0, ou seja, se a soma atinge um valor limite , a unidade de processamento dispara um pulso de saída. A comparação de *net* com o limiar μ é realizada pela função de Heaveside (função de escada) θ(x)=1 se x $\geq$0 e θ(x)=0 caso contrário.

$$y = \theta(\sum_{j=1}^{n} w_jx_j - \mu)$$

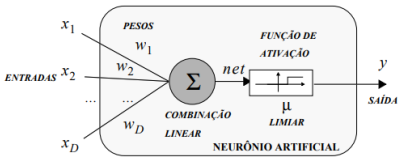

Figura 2: Modelo de um neurônio de McCulloch e Pitts . Fonte: Rauber (2005).

A partir desse modelo, houveram diversas funções de ativação para as unidades de processamento da rede. Klein & Martins (2006) e Rauber (2005)

2.3 Funções de Ativação

Atualmente as principais funções são:

- Função linear: Gera resultados de saída idênticos aos valores do potencial de ativação u, a expressão que representa matematicamente essa abordagem é:

$$g(u) = u$$

- Função Degrau: Aplicação da função degrau terá saída 1 se o potencial de ativação do neurônio for maior ou igual a zero, senão, terá valores nulos:

$$g(u) = \begin{cases} 1, se\ u \geq 0 \\ 0, se\ u < 0 \end{cases}$$

- Função Degrau Bipolar: Produz uma saída y para valores de entrada maiores que zero e uma saída −y para valores de entrada menores ou iguais a zero.

$$g(u) = \begin{cases} 1, se\ u \geq 0 \\ -1, se\ u < 0 \end{cases}$$

- Função Sigmóide: Possui apenas um máximo e mínimo, seu valor tende para zero quando a entrada tende para $-\infty$ e tende para um quando a entrada tende para $+\infty$.

$$Y(x) = \frac{1}{1 - e^{-\alpha x}}$$

x=Entrada. α=Determina a suavidade da curva.
Rauber (2005) e SANTOS et al. (2018)

2.4 Perceptron de Camada Simples

O perceptron simples é capaz de classificar entre duas classes que são linearmente separadas, ele utiliza do método de aprendizado supervisionado. A cada padrão de entrada apresentado à rede faz-se uma comparação entre a saída desejada e a calculada, tal comparação define o erro da resposta atual, os pesos então são recalculados para minimizar o fator de erro, ou seja, a cada etapa de treinamento faz-se pequenos ajustes nos pesos para que se chegue mais perto da solução desejada. Além disso, a sua arquitetura consiste numa rede de propagação para frente (feedforward) de camada simples, que consiste numa camada de entrada diretamente conexa a um ou mais neurônios que vão provocar a resposta de saída.Rauber (2005) e SANTOS et al. (2018)

Basicamente a ideia da tarefa de um neurônio artificial perceptron simples é a de classificação, que é realizada de acordo com a função de ativação: ϕ(z). Como apresentado no tópico 2.3 do artigo, há variedades diversas de função de ativação, no entanto, para conceituar o funcionamento do perceptron simples será utilizada a função degrau bipolar, a qual apresenta classificação binária, considerando dois valores, 1 e -1.

A função de ativação basicamente representa o disparo do neurônio, para isso ela necessita de uma combinação linear z com os sinais de entrada, x, e o vetor de pesos correspondente, w.

$$z = w_1x_1 + w_2x_2 + ... + w_nx_n$$

$$z = \sum_{i=0}^{n} x_i w_i = w^t x$$

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

Após realizar a combinação linear z, utiliza-se a função de ativação ϕ(z), que consiste na função degrau bipolar. Nesse caso, se o resultado for maior que o limiar θ=0, será atribuído o valor 1, caso contrário, -1.

$$\phi = \begin{cases} 1, se\ z \geq 0 \\ -1, caso\ contrário \end{cases}$$

Quando a saída é maior que zero, será disparado um sinal para a próxima camada da rede neural, caso contrário nenhum sinal será emitido.

Em particular, o modelo Perceptron simples, só pode ser aplicado em problemas lineares. A fim de ajustar o modelo, tem-se o conceito de bias. Basta imaginar o bias como uma taxa de ajuste constante, que ajuda a manter a linearidade da função.

Ao treinar a rede neural, tem-se um conjunto de dados com as classes separadas e pronta para o treinamento.

Considerando a regra de aprendizagem Δw_j, atualiza-se o valor de cada peso w_j, no vetor de pesos, w. Para definir formalmente a função de atualização dos valores do vetor de pesos, tem-se:

$$W_j = W_j + \Delta W_j$$

Nesse contexto, cada w_j, será atualizado considerando a regra de aprendizagem, Δw_j, definida como:

$$\Delta W_j = \eta\left(Y^{(i)} - \hat{Y}^{(i)}\right)X_j^{(i)}$$

η é a taxa de aprendizado (geralmente uma constante entre 0.0 e 1.0). y(i) é o valor da verdadeira classe para cada exemplo de entrada e ŷ(i) é o valor predito para cada entrada. Importante notar que todos os pesos do vetor de pesos, w, serão atualizados simultaneamente.

Utiliza-se a transposta de w, por ser apenas uma forma de escrever a multiplicação das matrizes. Portanto, como a entrada da rede z, é "esmagada" em uma saída binária (-1 ou 1) e então, pode ser usada para identificar entre duas classes linearmente separadas.Chrysostomo (2022)

2.5 Aplicações de Neurônios Artificiais

Os neurônios artificiais, ou melhor, as redes neurais artificiais apresentam aplicações diversas atualmente, tais como:

- Classificação de padrões: As redes podem aprender a diferenciar por exemplo a voz de pessoas, classificar caracteres e músicas, e realizar reconhecimento facial.

- Aproximação de funções: Capacidade de aprender qual é uma determinada função matemática que relaciona os dados de entrada aos dados de saída, como exemplo disso, uma rede é capaz de relacionar o timbre de um instrumento musical à outro e realizar a conversão entre eles.

- Segmentação em classes: RNAs possuem a habilidade de separar um conjunto de dados em grupos diferentes baseados em algum critério de semelhança entre eles, sendo assim, as redes podem encontrar padrões que seriam totalmente invisíveis aos humanos.

- Predição de séries temporais: A partir de de um conjunto de dados fornecidos anteriormente à rede, ela tem a capacidade de predizer novos dados futuros, tal aplicação é bastante relacionada por exemplo ao mercado financeiro ao tentar prever a queda ou subida de ações.Silva & Junior (2017)

3 Desenvolvimento

De acordo com os princípios teóricos estabelecidos anteriormente sobre os conceitos de neurônios e redes neurais artificiais, buscou-se implementar um algoritmo básico de aprendizado de um perceptron de camada simples em linguagem C.

O algoritmo escolhido a ser implementado apresenta a função de classificação de padrões. Basicamente, buscou-se realizar o aprendizado da rede para que ela pudesse diferenciar uma maça de uma laranja através da diferença entre seus pesos e pH. Para realizar o teste do algoritmo, utilizou-se de um banco de dados de entradas e saídas referente à classificação de cada fruta em relação à seu peso e pH. Para detalhamento técnico utilizou-se apenas 6 amostras, porém, o programa final apresenta 30 amostras.

ENTRADAS	PESO(g)	pH
x1	113	6,8
x2	122	4,7
x3	107	5,2
x4	98	3,6
x5	115	2,9
x6	120	4,2

SAÍDAS	FRUTA
y1	Maçã
y2	Laranja
y3	Maçã
y4	Maçã
y5	Laranja
y6	Laranja

Com base nas informações contidas nas tabelas o perceptron deverá ser capaz de identificar a reta que separa as classes (maçãs e laranjas).

No programa desenvolvido, os dados de entrada e saída, peso, pH e classe (fruta), foram armazenados em vetores de tamanho 30. No vetor de saída, o valor -1 representa as maçãs e o valor 1 representa as laranjas

Inicialmente, atribuiu-se o valor -1 ao bias, inicializou-se o vetor de pesos (W), de tamanho 3, com valor 0 e foi definida a taxa de aprendizagem como 0.1. Ao entrar na etapa de treinamento, determina-se a variável da saída do neurônio, Yr, que é calculada através da função de ativação. Para o calculo da função de ativação é necessário também o calculo da combinação linear das entradas e pesos, que consiste na soma do bias com os dados de entrada, multiplicados pelo seus respectivos pesos (W). Agora, na função de ativação, caso o resultado da combinação linear seja menor que zero, a saída do neurônio (Yr) será -1, caso a combinação linear tenha valor positivo, a saída do neurônio será 1. Desse modo, é possível calcular o erro (e) do neurônio caso tenha, o erro consiste na diferença entre a saída desejada e a saída do neurônio:

Y = -1:

$$Yr = \begin{cases} -1 & e=Yr-Y=-1-(-1)=0 \\ 1 & e=Yr-Y=-1-(+1)=-2 \end{cases}$$

Y = 1:

$$Yr = \begin{cases} -1 & e=Yr-Y=+1-(-1)=2 \\ 1 & e=Yr-Y=+1-(+1)=0 \end{cases}$$

Sendo assim, com o erro, é necessário a atualização do valor dos pesos, para isso basta utilizar a equação abaixo:

$$w_j = w_j + \eta e x_j$$

Na equação, η é a taxa de aprendizagem, e é o erro e x_j é a entrada

Após completar as etapas definidas, o perceptron termina um ciclo de treinamento, denominado época e então, se necessário inicia um novo. Geralmente é necessário uma determinada quantidade de épocas para que o perceptron possa aprender a diferenciar as classes.

Quando o vetor de erros apresentado na saída do programa apresentar todos valores 0, significa que o neurônio aprendeu a resolver o problema, ou seja, encontrou a reta que separa as classes.

O algoritmo base utilizado para desenvolver o programa foi:

```
1  Inicialização:
2  Faça w = [0]
3  Ativação e treinamento:
4  for k=1 a q do
5      i. Apresente o vetor de entradas Xb (k) ao
         perceptron
6      ii. Calcule a saída do neurônio Yr (k)
7      iii.Calcule o erro: e(k) = Yr (k) - y(k)
8      iv.Faça W (k) = W(k) + eta*e(k)*Xb(k)
9  end
10 Repita o passo de ativação e treinamento até que e(k) =
     0 para todo X(k)
```

Onde:
eta = taxa de aprendizado (0<eta<1)
Yr(k) = 1 se X(k) pertence a Classe 1, e Yr(k) = -1 se X(k) pertence a classe 2

O código final do perceptron está destacado na figura 3:

4 Resultados e Conclusões

O neurônio destacado na figura 3 foi capaz de realizar a separação das classes a partir da realização de 35 épocas o tempo médio do programa para realizar o calculo da reta que separa as classes é de 0.000102 segundos. Os gráficos da figura 4 apresentam o processo de treinamento do perceptron:

Nesta seção estudamos o artigo girando em torno dos resultados obtidos dele. Apresentamos anteriormente no decorrer deste artigo ordens matemática, estudos base que formaram o que hoje conhecemos do neurônio artificial, gráficos que apresentaram os resultados do algoritmo o qual introduzimos na prática um neurônio artificial.

```c
#include <stdio.h>
//Calcula a função de ativação(função sinal ou degrau bipolar)
float FuncaoAtivacao(float x){
 if (x<0)
   return -1;
  else
   return 1;
}
int main(void) {
//Define o número de epocas
int numEpocas = 35;
//Define o número de amostras
int q=30;
//Define as entradas do neurônio
int Peso[30] = {113, 122, 107, 98, 115, 120, 131, 103, 96, 140, 137, 112,
106, 126,144,152,139,111,101,141,157,161,135,129,154,100,174,162,148,159};
float pH[30] = {6.8, 4.7, 5.2, 3.6, 2.9, 4.2,
6.2,4.2,3.9,7.2,2.5,6.5,2.7,7.1,2.4,5.7,6.5,6.4,3.4,2.2,7.7,4.3,5.8,3.7,2.
8,6.1,4.7,3.4,7.6,3.1};
//Define a saída desejada do neurônio
int Y[30] = {-1, 1, -1, 1, 1,
1,-1,1,1,-1,1,-1,1,-1,1,1,-1,-1,1,1,-1,1,1,-1,1,1,-1,1};
//Define o vetor de erros, o bias, a taxa de aprendizado (n) e o vetor de
pesos das entradas e do bias
float e[30],bias = -1,n = 0.1,W[3]={0,0,0};
for (int i = 0; i < numEpocas; i++)
  for (int k = 0; k < q; k++){
    float Xb[3]={Peso[k],pH[k],bias},Yr,Somatorio;
    Somatorio=Peso[k]*W[0]+pH[k]*W[1]+bias*W[2];
    Yr = FuncaoAtivacao(Somatorio);
    e[k] = Y[k] - Yr;
    for(int l=0;l<3;l++){
      W[l]+=e[k]*n*Xb[l];
    }
  }
  printf("Vetor de erros (e) = [ ");
  for (int h = 0; h < 30; h++)
  printf ("%.2f ", e[h]);
  printf("]\n");
  printf ("Vetor de pesos (W) = [ %f , %f , %f ]\n", W[0], W[1], W[2]);
  return 0;
```

Figura 3: Código do Perceptron. Fonte: Autoria própria

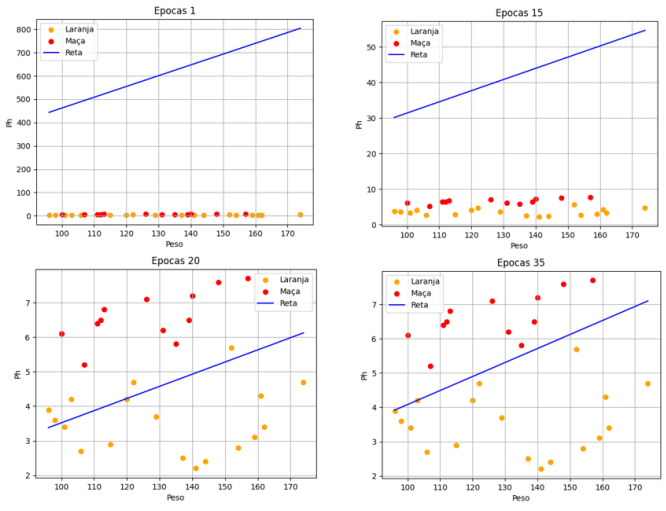

Figura 4: Etapas de treinamento. Fonte: Código gerado pelo ChatGPT

A criação da estrutura do algoritmo foi o momento o qual mais foi necessário trabalho dos alunos. Em conjunto, buscou-se hipóteses da melhor estrutura a fim de poder encaixá-la segundo os requisitos de um neurônio artificial, tendo de uma forma breve mas sem sacrificar partes essenciais da estrutura para, por fim, poder apresentá-la neste artigo.

Foram sendo apresentados os modelos, fórmulas, estruturas matemáticas que são necessárias para a base de um neurônio artificial, como também gráficos e imagens que elucidam as ideias apresentadas no artigo. Por fim, terminando com o algoritmo que introduz a prática de toda nossa teoria.

O estudo de um neurônio artificial e RNA como um todo estão relacionados a diversas disciplinas dentro da ciência da computação. Falamos de ambos pois neurônios e redes estão interligado diretamente. Vale a pena destacar algumas dessas disciplinas e sua conexão ao tema do artigo:

1. Arquitetura de Computadores: Trata da estrutura e organização dos sistemas computacionais. Entender como os sistemas computacionais podem ser projetados para executar cálculos de forma eficiente;

2. Matemática Discreta: Envolve conceitos como álgebra linear, cálculo, probabilidade e estatística, i.e, conhecimentos matemáticos que são usados para modelar e descrever os neurônios artificiais, as funções de ativação, os algoritmos de aprendizado e as operações realizadas;

3. Fundamentos de Programação: Essencial na implementação de algoritmos e estruturas de dados utilizados na construção do neurônio artificial. Compreender conceitos como variáveis, estruturas condicionais, loops e funções;

4. Cálculo: Usado para otimização de parâmetros, ajuste de pesos e atualização dos modelos de aprendizado. Conceitos como derivadas e gradientes são fundamentais para os algoritmos de aprendizado, como o backpropagation.

Essas são algumas das relações entre o neurônio artificial e consequentemente as RNA, e as disciplinas mencionadas. Cada disciplina contribui com seus conceitos e técnicas para o estudo, desenvolvimento e aplicação dos neurônios artificiais.

Trabalho realizado pelos alunos:
Caio: Desenvolvimento do algoritmo e do programa;
Davi: Modelo de McCulloch e Pitts e funções de ativação;
Isaac: Perceptron e correlação com as disciplinas de CCO;
Raphael: Neurônio biológico e redes neurais;
Tiago: Desenvolvimento do algoritmo e do programa;

Referências

Chrysostomo, N. (2022). Neurônios artificiais – modelo perceptron. Disponível em:https://blog.gft.com/br/2022/02/15/neuronios-artificiais-modelo-perceptron/; Acessado em:16/06/2023.

Klein, C. F. Á. & Martins, J. F. J. (2006). Implementação de rede neural em hardware de ponto fixo.

Rauber, T. W. (2005). Redes neurais artificiais. *Universidade Federal do Espírito Santo*, 29.

SANTOS, F. K. M. d. et al. (2018). Redes neurais artificiais e suas aplicabilidades: modelagem do valor da temperatura máxima na cidade de castanhal.

Silva, A. & Junior, V. B. d. S. (2017). AplicaÇÕes e benefÍcios obtidos atravÉs das redes neurais artificias (rna). Disponível em:https://www.facima.edu.br/instituto/revista/arquivos/ano2/revista_facima_ano_2_aplicacoes_beneficios.pdf; Acessado em:10/06/2023.