COMPARAÇÃO ENTRE ART FUZZY E BACKPROPAGATION PARA IDENTIFICAÇÃO DE ERROS DE DIGITAÇÃO

André Ananias Barreto[1]
Fahme Alves Salim Junior[1]
James Clauton[2]

RESUMO

O trabalho descreve a implementação de duas Redes Neurais utilizando os algoritmos de *backpropagation* e *ART Fuzzy* para a resolução de um problema de "tema livre", cujo mesmo se diz respeito à identificação de erros de digitação recebidos por um *chatbot*. O objetivo é descobrir qual das abordagens de Rede Neural oferece respostas mais coerentes sobre a possibilidade de duas palavras de igual tamanho, com diferença de apenas uma letra, serem resultado ou não de um erro de digitação. Após a implementação e testes, a Rede Neural *ART Fuzzy* mostrou resultados melhores do que a rede com *backpropagation*.

Palavras-Chave: Redes neurais; Retropropagação, *ART Fuzzy*, Redes da família *ART*.

[1] Acadêmicos do 9º termo do curso de Engenharia da Computação do Centro Universitário Católico Salesiano Auxilium – UniSALESIANO de Araçatuba.
[2] Orientador

ABSTRACT

The work describes the implementation of two neural networks that use backpropagation and ART Fuzzy algorithms to solve a "free theme" problem, the chosen being identifying typos received by a chatbot. The goal is to find out which of the Neural Network approaches offers more coherent answers about whether or not two identical words, with a difference of just one letter, are the result of a typo. After implementation and testing, the Neural ART Fuzzy Network showed better results than a backpropagation network.

Keywords: Neural networks; Backpropagation, ART Fuzzy, ART family networks.

Introdução

Ao decorrer da matéria de Sistemas Inteligentes, foi proposto pelo professor James o desenvolvimento de um projeto de inteligência artificial que resolveria um problema (sendo esse problema hipotético ou não) com escolha livre de linguagem, para a resolução do mesmo foi decretado o uso do algoritmo de *Backpropagation* como Núcleo da mesma. Posteriormente, foi pedido um novo trabalho, resolvendo o mesmo problema, porém desta vez utilizando uma rede neural da família ART.

Como bem sabemos, *Backpropagation* é um modelo de Redes neurais que visa como resultado a diminuição do Erro Proposto.

Há quem diga que o *Backpropagation* é o algoritmo mais importante da história das Redes Neurais pois sem o mesmo seria impossível treinar redes de aprendizagem profunda como vemos atualmente.

A mesma é composta de duas fases:

- O *Forward pass* é onde as entradas são passadas através da rede por seus neurônios e também são anunciadas as previsões de saída.

- O *Backward pass* é onde são calculados os gradientes da função perda (chamada de previsão), esse gradiente é usado recursivamente na *chain rule*, atualizando assim os pesos por toda a rede.

As redes da família ART por sua vez, são redes capazes de assimilar padrões sem perder conhecimento já obtido anteriormente. A ART Fuzzy permite a entrada de valores analógicos, não apenas binários. Seu funcionamento, é a partir de reconhecimento de padrões conhecidos, para cada valor, será devolvida a posição das entradas aprendidas que for mais parecida com o valor dado.

O problema em questão foi escolhido tendo como base o uso de *chatbots*, estes são, em suma, robôs programados para responder mensagens de um usuário humano, um dos maiores problemas neste procedimento é a detecção de equívocos ortográficos, se a palavra escrita pelo usuário foi propositalmente escrita daquela forma, ou se o mesmo trocou algum caractere durante a sua digitação.

Este trabalho busca descobrir se uma pequena Rede Neural pode oferecer respostas coerente sobre a possibilidade de duas palavras de igual tamanho, com diferença de apenas uma letra, são resultado ou não de um erro de digitação.

A linguagem escolhida para o projeto foi *phyton* 3 devido a experiência prévia nessa linguagem e por conta da facilidade para implementação de Redes Neurais.

No caso, o algoritmo programado deve fazer uma validação de erro tendo em base a distância de cada tecla no teclado, quanto mais próximo, maior a chance do usuário ter errado, e quanto mais longe, menor as chances de erro, para fazer isso possível, utilizamos de técnicas evidenciadas nos tópicos abaixo.

Metodologia

Cálculo de saída

Para realização do cálculo, duas palavras são submetidas à lógica. As entradas das redes são então calculadas a partir das palavras.

A Equação 1 determina a primeira entrada, resultando em um número que começa em 0,5 para palavras com duas letras e se aproxima de um, quanto maior o

tamanho da palavra em caracteres. Essa função foi escolhida pois quanto mais caracteres uma palavra tem, maiores as chances de um erro de digitação.

A segunda entrada é baseada na posição das palavras no teclado e será 1 se as letras que diferem nas palavras inseridas então lado a lado no teclado, e será zero caso contrário.

A primeira e a segunda entrada são então submetidas à Rede Neural Artificial.

$$Primeira\ entrada = \frac{tamanho - 1}{tamanho} \quad (1)$$

Como exemplo, diga-se que ao tentar escrever a palavra “soja” o indivíduo escreveu “soka”, a primeira entrada será o resultado da Equação 1 com tamanho sendo 4(caracteres), resultando em 0,75. A segunda entrada será 1 já que o caractere “k” está ao lado do “j” no teclado.

Foram desenvolvidas duas Redes Neurais, uma utilizando *Backpropagation* e outra utilizando *ART Fuzzy*.

Backpropagation

A rede deve identificar, a partir de duas palavras com diferença de apenas uma letra, se a diferença se trata de um erro de digitação, retornando um número entre 0 e 1, representando a chance de o erro ser um erro cometido na hora de editar.

A implementação da rede neste caso, apresenta dois neurônios na camada escondida e um na camada final.

A linguagem de desenvolvimento utilizada foi *python* 3, como não se podia utilizar bibliotecas complexas a única utilizada foi a “*math*” para realizar a exponenciação e nada mais.

Cada neurônio - e consequentemente a rede - pode receber duas entradas e retorna um valor entre 0 e 1, para isso foi usada a Equação 2 como sigmóide(função

de ativação), sendo λ a inclinação da função, *s* a saída intermediária do neurônio e e o número de Neper. A saída intermediária é a somatória dos pesos multiplicados pelas entradas mais um peso multiplicado pelo bias.

$$y = \frac{1}{1 + e^{(-\lambda s)}} \quad (2)$$

Essa lógica foi implementada em uma classe chamada Neuron em uma função chamada “run”. Também foi implementada a função “update” que calcula o erro e atualiza os pesos.

Para que o erro se propague pela rede foi criada uma classe Network, que guarda cada Neuron e chama a função update para cada um de trás para frente, se repetindo até que o erro alcance o limite máximo pré estabelecido.

As lógicas que não fazem parte da Rede Neural foral colocadas no arquivo principal(“*main.py*”) que também chama a classe Network realizando treino e teste.

Os dados de treino foram simples sendo mostrados na Figura 1. Os dois primeiros valores são as duas entradas e o último a saída esperada. Observa-se que eles foram preparados de forma a dar mais peso para a segunda entrada, visto que após múltiplos testes, esta forma mostrou melhores resultados.

Figura 2 - Dados de treinamento *backpropagation*

```
training_data = [[1, 1, 1.00],
                 [1, 0, 0.25],
                 [0, 0, 0.00],
                 [0, 1, 0.75]]
```

Fonte: elaboração própria, 2020.

ART Fuzzy

A rede recebe dois padrões possíveis e retorna o padrão mais parecido com a entrada dada.

A linguagem de desenvolvimento utilizada foi também o *python* 3, e como não se podia utilizar bibliotecas complexas a única utilizada foi a "*numpy*" para realizar a criação de uma matriz com 1 em todas as posições, cuja utilidade é mostrada abaixo.

Para o treinamento é preciso realizar o complemento dos padrões dados, foi criada uma função *create_complement* que faz isso para toda a matriz de entrada.

Após isso, é criada uma matriz semelhante em tamanho à matriz de entrada, porém contendo 1 em todas as posições.

Em seguida são criadas as categorias para cada entrada. A definição das categorias é mostrada pela Equação 3, sendo *alfa* chamado parâmetro de escolha.

$$T = \frac{|Entradas \cap Pesos|}{\alpha + |Pesos|} \quad (3)$$

A categoria de maior valor e menor índice é selecionada e passa pelo teste de vigilância, sendo *rho* o parâmetro de vigilância da rede, Equação 4.

$$\frac{|Entradas \cap Pesos|}{|Entradas|} \geq \rho \quad (4)$$

Se o resultado do teste for verdadeiro, a rede passa a atualizar os pesos, caso contrário uma nova categoria é escolhida.

A atualização dos pesos para cada linha, é mostrada na Equação 5, sendo *beta* a taxa de treinamento.

$$Pesos = \beta(Entradas \cap Pesos)(1 + \beta) \times Pesos \quad (5)$$

Os dados do treino são mostrados na Figura 2. Se a resposta escolher a primeira linha como semelhante, significa que a diferença entre as palavras provavelmente é um acidente, caso a segunda linha seja escolhida, as palavras têm grandes chances de realmente serem palavras diferentes.

Figura 2 - Dados de treinamento *ART Fuzzy*

```
data = [[1, 1],
        [0.3, 0]]
```

Fonte: elaboração própria, 2020.

Resultados e Discussão

Após treinamento com 863 iterações no *Backpropagation* e um treinamento mais veloz com *ART Fuzzy*, foram submetidos quatro testes aos modelos.

O primeiro entre as palavras “soja” e “soka”, o *Backpropagation* mostra um resultado alto de 0.65 de chances de ser um erro de digitação, condizente com a realidade, porém a certeza apresentada pela rede *ART Fuzzy* é ainda maior. Ver Figuras 3 e 4.

Figura 3 - Saída do Teste 1 com *backpropagation*, entre as palavras “soja” e “soka”

```
Teste: soja vs soka
Input 1: 0.75, Input 2: 1 = Output: 0.6537397105903469
```

Fonte: elaboração própria, 2020.

Figura 4 - Saída do Teste 1 com *ART Fuzzy*, entre as palavras “soja” e “soka”

```
Teste: soja vs soka
Input 1: 0.75, Input 2: 1
Output: 0.8333333333333333([1, 1, 0, 0])
```

Fonte: elaboração própria, 2020.

O segundo teste, entre as palavras “soja” e “soda”, o *Backpropagation* mostra um resultado baixo de 0.34 de chances de ser um erro de digitação, também condizente com a realidade visto que a palavra “soda” existe. O *ART Fuzzy* também mostra uma alta certeza indicando a categoria de palavras diferentes. Ver Figuras 5 e 6.

Figura 5 - Saída do Teste 2 com *backpropagation*, entre as palavras “soja” e “soda”

```
Teste: soja vs soda
Input 1: 0.75, Input 2: 0 = Output: 0.3412521946958812
```

Fonte: elaboração própria, 2020.

Figura 6 - Saída do Teste 2 com *ART Fuzzy*, entre as palavras “soja” e “soda”

```
Teste: soja vs soda
Input 1: 0.75, Input 2: 0
Output: 0.7380952380952381([0.3, 0, 0.7, 1])
```

Fonte: elaboração própria, 2020.

No terceiro teste, com a palavra propositalmente grande “pneumoultramicroscopicossilicovulcanoconiótico”, o *Backpropagation* mostra um resultado relativamente alto de 0.66 de chances de ser um erro de digitação, também condizente com a realidade, porém novamente a rede *ART Fuzzy* mostra uma certeza muito maior. Ver Figuras 7 e 8.

Figura 7 - Saída do Teste 3 com *backpropagation*, a palavra “pneumoultramicroscopicossilicovulcanoconiótico” e a mesma com erro no último caractere

```
Teste: pneumoultramicroscopicossilicovulcanoconiotico vs pneumoultramicroscopicossilicovulcanoconioticp
Input 1: 0.9782608695652174, Input 2: 1 = Output: 0.665845727413164
```

Fonte: elaboração própria, 2020.

Figura 8 - Saída do Teste 3 com *ART Fuzzy*, a palavra “pneumoultramicroscopicossilicovulcanoconiótico” e a mesma com erro no último caractere

```
Teste: pneumoultramicroscopicossilicovulcanoconiotico vs pneumoultramicroscopicossilicovulcanoconioticp
Input 1: 0.9782608695652174, Input 2: 1
Output: 0.9420289855072463([1, 1, 0, 0])
```

Fonte: elaboração própria, 2020.

Por fim, o quarto teste, entre as palavras “james” e “jsmes” é semelhante ao primeiro e mostra um resultado relativamente alto de 0.65 de chances de ser um erro

de digitação para o *Backpropagation*, mostrando a consistência do modelo. O *ART Fuzzy,* apresenta novamente uma grande certeza com 0,85. Ver Figuras 9 e 10.

Figura 9 - Saída do Teste 4 com *backpropagation*, entre as palavras “james” e “jsmes”

```
Teste: james vs jsmes
Input 1: 0.8, Input 2: 1 = Output: 0.6564977246289162
```

Fonte: elaboração própria, 2020.

Figura 10 - Saída do Teste 4 com *ART Fuzzy*, entre as palavras “james” e “jsmes”

```
Teste: james vs jsmes
Input 1: 0.8, Input 2: 1 = Output: 0.6564977246289162
```

Fonte: elaboração própria, 2020.

Conclusão

Foi possível criar um modelo de Rede Neural consistente que ajuda a definir se duas palavras com diferença de um caractere sofreram erro de digitação ou são realmente palavras diferentes com ambos os modelos.

Porém a rede *ART Fuzzy*, precisou de menos processamento e exemplos de teste, e ainda assim mostrou resultados mais certeiros.

Futuros trabalhos podem realizar a inclusão dos modelos no fluxo de *Chatbots* para avaliar sua relevância prática, favorecendo o *ART Fuzzy* que teve melhores resultados.

Referências Bibliográficas

Conteúdo dado em aula pelo Professor James. Unisalesiano.

<Capítulo 15 - Algoritmo Backpropagation Parte 2 - Treinamento de Redes Neurais>
Acesso em 12 de Maio de 2020.

<https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_adaptive_resonance_theory.htm>
Acesso em 06 de Junho de 2020.