Universidade Federal do Maranhão
Programa de Pós Graduação Em Ciência da Computação

Aluno
Carlos Eduardo Nascimento Cajado

Professor
Prof. Dr. Areolino de Almeida Neto

RECONHECIMENTO DE FONEMAS

São Luís, MA
Maio - 2024

1. INTRODUÇÃO

1.1 Contexto

De acordo com a definição de HAYKIN, uma rede neural é um sistema projetado para imitar a maneira como o cérebro realiza tarefas específicas, geralmente implementado com componentes eletrônicos ou por meio de algoritmos computacionais. Para alcançar um bom desempenho, é essencial que as redes neurais utilizem uma interconexão massiva de células computacionais simples, chamadas de "neurônios" ou unidades de processamento (HAYKIN, 2001) [2].

Ademais, os neurônios artificiais são organizados em camadas, onde cada camada é responsável por diferentes níveis de abstração e processamento dos dados. As redes neurais aprendem ajustando as conexões entre os neurônios através de algoritmos de treinamento, como o backpropagation, que minimiza os erros na saída. Com o treinamento adequado, as redes neurais podem reconhecer padrões complexos, realizar classificações e fazer previsões com alta precisão.

1.2 Problema

Neste estudo, utilizaremos a rede neural MLP (Multilayer Perceptron) *backpropagation*, visando aprender a reconhecer as sílabas fonéticas das palavras “DIREITA” e “ESQUERDA”.

1.3 Objetivos

- Desenvolver e Treinar a Rede Neural MLP.
- Analisar a Precisão do algoritmo.
- Verificar o reconhecimento das palavras.
- Determinar o melhor ponto de parada do treinamento.

1.4 Configuração do Ambiente

Para desenvolvimento do projeto, temos as seguintes configurações:

1.3.1 Software:

- Windows 11
- Audacity
- Matlab

1.3.2 Hardware:

- Processador: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
- Memória RAM: 16,0 GB (utilizável: 15,9 GB)

2. BASE DE DADOS

2.1 pré-processamento:

Primeiramente, utilizando o aplicativo "Audacity", é fundamental segmentar as sílabas fonéticas (DI, REI, TA, ES, QUER, DA) para serem utilizadas no treinamento. Nesse caso, temos 73 amostras da palavra "esquerda" e 58 amostras de "direita" como conjunto totais.

Dividindo o *Database*:

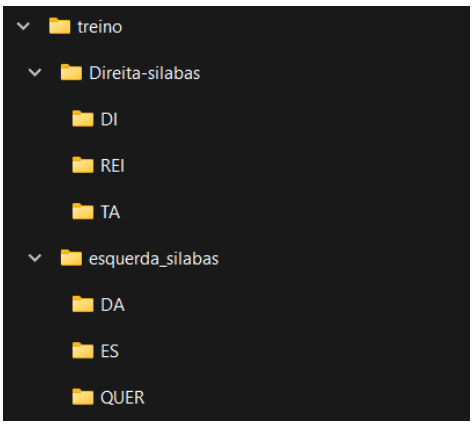
Treino: 75%

Validação: 15%

Teste: 10%

Temos a seguinte divisão de diretórios, a exemplo, para treino:

Imagem 1: Particionamento das Sílabas



Autor: próprio

Essa divisão se repete para pasta de validação e teste.

2.2 Lendo os arquivos de áudio no matlab :

Utilizando o software MATLAB, serão desenvolvidos vários scripts de pré-processamento. Cada pasta de áudios, incluindo as de treino, teste e validação, terá um script dedicado que seguirá as etapas abaixo:

1. Leitura dos caminhos de cada sílaba.
2. Processar cada faixa de áudio utilizando a biblioteca “audioRead”.
3. Calcular a Transformada Rápida de Fourier (FFT) da primeira coluna dos dados de áudio, retornando a magnitude, abs (valor absoluto).
4. Atribuir a cada sílaba , utilizado “mear”, o valor da média dos elementos.
5. Salvar variáveis do MatLab, em arquivo com extensão mat.

A exemplo temos, na imagem 2, um trecho correspondente ao arquivo treino. Visualizamos os passos para a fonema “DI”.

Imagem 2: Trecho pré-processamento de treino.

```
% Diretório para cada sigla
diretorio_DI = 'treino\Direita-silabas\DI';
diretorio_REI = 'treino\Direita-silabas\REI';
diretorio_TA = 'treino\Direita-silabas\TA';

% Carregando audios e inicializando células para armazenar dados de FFT
audio_di = dir(fullfile(diretorio_DI, '*.wav'));
audio_rei = dir(fullfile(diretorio_REI, '*.wav'));
audio_ta = dir(fullfile(diretorio_TA, '*.wav'));


dados_audio_di = cell(1, numel(audio_di));
dados_audio_rei = cell(1, numel(audio_rei));
dados_audio_ta = cell(1, numel(audio_ta));

N_partes = 128;
% áudios DI
for i = 1:numel(audio_di)

     arquivo_atual = fullfile(diretorio_DI, audio_di(i).name);
     [audio_data, ~] = audioread(arquivo_atual);
     fft_data = abs(fft(audio_data(:,1)));

    % Dividindo fft_data em partes
    tamanho_parte = floor(numel(fft_data) / N_partes);
    partes_fft = reshape(fft_data(1:N_partes*tamanho_parte), tamanho_parte, N_partes);

    % Calculando a média de cada parte
     dados_audio_di{i} = mean(partes_fft) ;
end
```

Autor: próprio

O vetor resultante da FFT será dividido em “N_partes” visando reduzir a dimensionalidade dos dados de áudio e extrair características representativas de cada segmento do espectro de frequências.

3 TREINAMENTO

Partindo da configuração:

- As correspondências das classes foram criadas de modo que cada fonema corresponde a uma classe numerada de 1 a 6. A Tabela 1 mostra a relação entre cada fonema e sua

respectiva classe.

Tabela 1: correspondências das classes.

Fonema	Número classe correspondente
“DI”	1
“REI”	2
“TA”	3
“ES”	4
“QUER”	5
“DA”	6

Autor: próprio

- Configuração da rede.

No planejamento, foi definido inicialmente o uso de 6 neurônios, um para cada fonema, com um número reduzido de neurônios na camada oculta. Esse número será aumentado conforme as dificuldades encontradas durante o treinamento, visando encontrar um equilíbrio adequado.

```
% Parâmetros da rede neural
numClasses = 6;         % Número de classes na camada de saída
hiddenLayerSize = 4;    % Número de neurônios na camada oculta
eta = 0.0001;            % Taxa de aprendizado

% Definir o número máximo de épocas
maxEpocas = 50000;
```

3.2 Treinamento:

Adicionado Atributos (X) e classes (Y). Seguimos:

```
dados_di = cat(1, dados_audio_di{:});
dados_rei= cat(1, dados_audio_rei{:});
dados_ta = cat(1, dados_audio_ta{:});

dados_es = cat(1, dados_audio_es{:});
dados_quer = cat(1, dados_audio_quer{:});
dados_da = cat(1, dados_audio_da{:});


%6 CLASSES; DI, REI, TA, ES, QUER, DA
X = [dados_di; dados_rei; dados_ta; dados_es;  dados_quer; dados_da];

Y = [1*ones(size(dados_di', 2), 1); 2*ones(size(dados_rei', 2), 1); 3*ones(size(dados_ta', 2), 1);
        4*ones(size(dados_es', 2), 1); 5*ones(size(dados_quer', 2), 1); 6*ones(size(dados_quer', 2), 1)];
```

O algoritmo de treinamento da rede neural segue um conjunto de passos bem

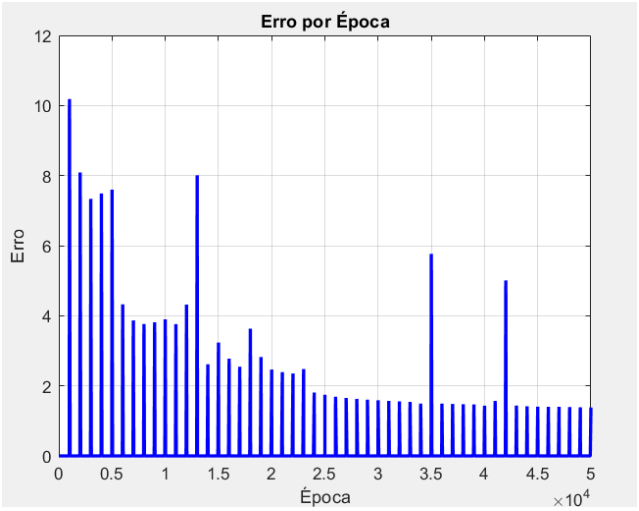
definidos, baseados nos estudos encontrados em [1].

1. **Gerando pesos e bias aleatórios**: gerando os pesos aleatórios conforme o número de neurônios.
2. **Cálculo da entrada da camada escondida**: Computa-se a entrada para a camada escondida utilizando os pesos e bias atuais.
3. **Cálculo da saída da camada escondida**: será Aplicado a função de ativação "sigmoide" à entrada da camada escondida para obter a saída.
4. **Cálculo da entrada da camada de saída**: Calcula-se a entrada para a camada de saída com base na saída da camada escondida e nos pesos e bias correspondentes.
5. **Cálculo da saída da rede neural**: Obtém-se a saída final da rede aplicando uma constante "K" à entrada da camada de saída. Nos nossos estudos a constante é fixada em 1.
6. **Cálculo do erro**: O erro "E" é uma comparação da saída prevista com a saída desejada (Y).
7. **Atualização dos pesos e bias da camada de saída**: Calculamos as variações dos pesos e bias entre a camada escondida e a camada de saída com base no erro "E'.
8. **Cálculo do erro retropropagado**:Onde o erro que deve ser propagado de volta para a camada escondida.
9. **Atualização dos pesos e bias da camada escondida**: Etapa onde as variações dos pesos e bias entre a camada de entrada e a camada escondida com base no erro propagado são atualizadas.
10. **Cálculo do erro quadrático médio (Eav)**: Calcula-se e registra-se o erro quadrático médio da rede para monitorar o desempenho ao longo das épocas.
11.

3.3 Saída da rede Neural:

Para plotar um gráfico podemos calcular o erro quadrático médio (Eav) da rede, obtido a cada 1000 épocas. Podemos notar com o gráfico, imagem 4, a queda do erro "Eav" conforme a contagem das épocas.

Imagem 4: erro quadrático médio (Eav) .



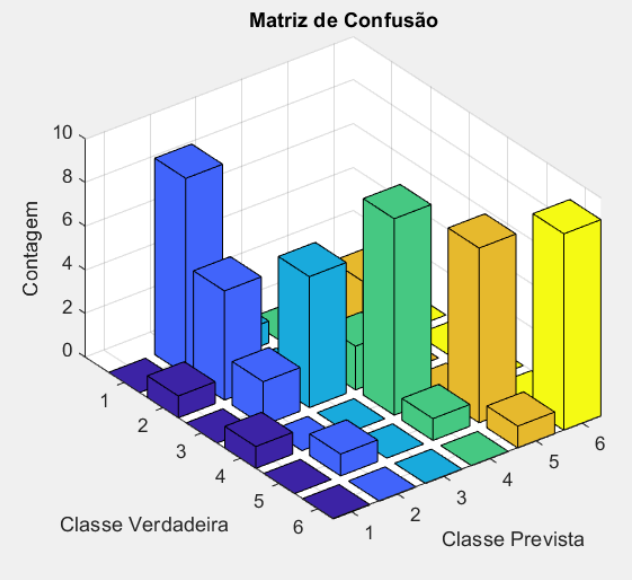
Autor: próprio

É evidente que conforme o número de épocas aumenta, o erro tende a diminuir, embora haja alguns picos na curva, possivelmente devido a uma taxa de aprendizado muito alta, entretanto, outras variações de taxa prejudicaram o treinamento, gerando valores inferiores aos de testes.

3.2 Validação:

Rodando o algoritmo de validação durante o treinamento, obtivemos diversas informações importantes, destacando-se a matriz de confusão, conforme mostrado na Imagem 3. Esta matriz revela o desempenho do classificador em reconhecer seis classes distintas. Observa-se que a Classe 4, 5e a Classe 6 foram classificadas corretamente na maioria das vezes, com 9 acertos cada, e poucas confusões. No entanto, há dificuldades notáveis em distinguir a Classe 1, que foi majoritariamente confundida com a Classe 2. A Classe 2 também apresenta confusões significativas, especialmente com a Classe 5. As demais classes possuem uma boa taxa de acertos. No geral, o modelo mostrou uma precisão aceitável em aproximadamente 70%, mas ainda há a necessidade de melhorar a distinção entre classes mais frequentemente confundidas.

Imagem 3: Matriz de confusão validação.



Autor: próprio

4 TESTE

Para o *script* de teste, serão passados os atributos X inéditos, ainda não vistos pela rede.

```
% Carregar os dados DIREITA!!
load('dados_audio_DI_teste.mat');
load('dados_audio_REI_teste.mat');
load('dados_audio_TA_teste.mat');

% Carregar os dados ESQUERDA!!
load('dados_audio_ES_teste.mat');
load('dados_audio_QUER_teste.mat');
load('dados_audio_DA_teste.mat');

dados_teste_di = cat(1, dados_audio_di_teste{:});
dados_teste_rei= cat(1, dados_audio_rei_teste{:});
dados_teste_ta = cat(1, dados_audio_ta_teste{:});

dados_teste_es = cat(1, dados_audio_es_teste{:});
dados_teste_quer = cat(1, dados_audio_quer_teste{:});
dados_teste_da = cat(1, dados_audio_da_teste{:});
```

4. 1 Classificando as amostras (TESTE):

O algoritmo de teste pode ser encontrado como pseudocódigo abaixo:

Algorithm 3 Processo de Teste e Avaliação da Rede Neural

Entrada: Dados de teste: $dados\_teste\_di, rei, ta.dados\_teste\_es, quer, da$
Saída: Classe prevista para cada amostra e contagem de acertos
▷ Passo 2: Calcular entrada da camada escondida
$net\_h\_teste \leftarrow Whi \times X\_teste^T + bias\_hi \times ones(1, size(X\_teste^T, 2))$
▷ Passo 3: Calcular a saída da camada escondida
$Yh\_teste \leftarrow logsig(net\_h\_teste)$
▷ Passo 4: Calcular entrada da camada de saída
$net\_o\_teste \leftarrow Woh \times Yh\_teste + bias\_oh \times ones(1, size(Yh\_teste, 2))$
▷ Passo 5: Calcular a saída da rede neural
$Ys\_teste \leftarrow net\_o\_teste$
▷ Calcular a classe prevista para cada amostra
$Ys\_mean \leftarrow mean(Ys\_teste)$
$classe\_prevista \leftarrow round(max(Ys\_mean, [], 1))$
$acertos \leftarrow 0$
for $i = 1$ **to** length($classe\_prevista$) **do**
    fprintf('Amostra %d: Valor Previsto = %d, Valor Real = %d', i, $classe\_prevista[i]$, $Y\_teste[i]$)
    if $classe\_prevista[i] == Y\_teste[i]$ **then**
        $acertos \leftarrow acertos + 1$
    end if
end for

Autor: próprio

4 .2 Saída arquivo teste:

Como saída, após alguns treinamento e seleção dos melhores pesos conseguimos 84,78% de acurácia nos testes. O *output* do terminal do matlab encontra-se na imagem 4.

Imagem 4: Output terminal teste (acurácia) .

```
Amostra 40: Valor Previsto = 6, Valor Real = 6
Amostra 41: Valor Previsto = 5, Valor Real = 6
Amostra 42: Valor Previsto = 4, Valor Real = 6
Amostra 43: Valor Previsto = 6, Valor Real = 6
Amostra 44: Valor Previsto = 6, Valor Real = 6
Amostra 45: Valor Previsto = 6, Valor Real = 6
Amostra 46: Valor Previsto = 6, Valor Real = 6
Acurácia: 84.78%
fx >>
```

Autor: próprio

5 RECONHECIMENTO DAS PALAVRAS

Criando uma função que recebe 3 sílabas, na ordem correta, e retorna a palavra caso seja reconhecida pela rede. encontramos o algoritmos com o pseudocódigo.

Algorithm 4 Função de Reconhecimento de Palavra

```
function RECONHECE_PALAVRA(silaba_1, silaba_2, silaba_3)
    palavra_reconhecida ← ”
                                ▷ Chamada da função Saida_rede para a primeira sílaba
    classe_prevista ← SAIDA_REDE(silaba_1)
                            ▷ Verifica a classe prevista e atualiza a palavra reconhecida
    if classe_prevista == 1 then
        DISP('Sílaba DI reconhecida')
        classe_prevista ← SAIDA_REDE(silaba_2)
        if classe_prevista == 2 then
            DISP('Sílaba REI reconhecida')   ▷ Chamada da função Saida_rede
para a terceira sílaba
            classe_prevista ← SAIDA_REDE(silaba_3)
            if classe_prevista == 3 then
                DISP('Sílaba TA reconhecida')
                palavra_reconhecida ← 'DIREITA'
            end if
        end if
    else if classe_prevista == 4 then
        DISP('Sílaba ES reconhecida')
        classe_prevista ← SAIDA_REDE(silaba_2)
        if classe_prevista == 5 then
            DISP('Sílaba QUER reconhecida')          ▷ Chamada da função
Saida_rede para a terceira sílaba
            classe_prevista ← SAIDA_REDE(silaba_3)
            if classe_prevista == 6 then
                DISP('Sílaba DA reconhecida')
                palavra_reconhecida ← 'ESQUERDA'
            end if
        end if
    else
        palavra_reconhecida ← 'PALAVRA NÃO RECONHECIDA'
    end if
    return palavra_reconhecida
end function
```

Autor: próprio

Reconhecendo as palavras:

Visando testar o reconhecimento de palavras pela união de três fonemas, utilizaremos um algoritmo dividido em duas partes. Na primeira parte, uma função utiliza uma rede neural previamente treinada para calcular a saída da rede, retornando a classe prevista para a entrada fornecida. Na segunda parte, são selecionadas amostras aleatórias de sílabas associadas às palavras "DIREITA" e "ESQUERDA", testando o reconhecimento dessas palavras por meio de uma função de reconhecimento e exibindo os resultados dos testes para cada amostra selecionada. Na imagem 5, temos a saída do teste.

Como saída temos:

Imagem 5: Output terminal teste (Reconhecendo palavras) .

```
Sílaba DI reconhecida
Sílaba REI reconhecida
Sílaba TA reconhecida
Teste DIREITA 1 - Saída: DIREITA
__________________
Sílaba DI reconhecida
Sílaba REI reconhecida
Sílaba TA reconhecida
Teste DIREITA 2 - Saída: DIREITA
__________________
Sílaba DI reconhecida
Teste DIREITA 3 - Saída:
__________________
Sílaba DI reconhecida
Sílaba REI reconhecida
Sílaba TA reconhecida
Teste DIREITA 4 - Saída: DIREITA
__________________
Sílaba DI reconhecida
Teste DIREITA 5 - Saída:
__________________
****************************

Teste ESQUERDA 1 - Saída:
__________________
Sílaba ES reconhecida
Sílaba QUER reconhecida
Sílaba DA reconhecida
Teste ESQUERDA 2 - Saída: ESQUERDA
__________________
Sílaba ES reconhecida
Sílaba QUER reconhecida
Sílaba DA reconhecida
Teste ESQUERDA 3 - Saída: ESQUERDA
__________________
Sílaba ES reconhecida
Sílaba QUER reconhecida
Sílaba DA reconhecida
Teste ESQUERDA 4 - Saída: ESQUERDA
__________________
Sílaba ES reconhecida
Sílaba QUER reconhecida
Sílaba DA reconhecida
Teste ESQUERDA 5 - Saída: ESQUERDA
__________________
```

Autor: próprio

Ou seja, nesses testes, observa-se que a palavra "DIREITA" foi reconhecida corretamente em 3 de 5 tentativas (3/5) e a palavra "ESQUERDA" foi reconhecida corretamente em 4 de 5 tentativas (4/5).

6 CONCLUSÃO:

Neste estudo, foi possível desenvolver e treinar uma rede neural MLP (Multilayer Perceptron) com o algoritmo de backpropagation para o reconhecimento das sílabas fonéticas das palavras "DIREITA" e "ESQUERDA". Através do ajuste de parâmetros iniciais, como a taxa de aprendizagem e o número de neurônios nas camadas de entrada e saída, a rede neural demonstrou ser capaz de realizar classificações com alta precisão. A análise dos resultados confirmou a eficácia do algoritmo no reconhecimento de fonemas, mostrando que a rede conseguiu aprender e generalizar para os dados de teste.

REFERÊNCIAS :

[1] ALMEIDA NETO, A. Rede Perceptron de Múltiplas Camadas.Slide PPGCC, 2024.

[2] HAYKIN, S. Neural networks: a comprehensive foundation. New Jersey: Prentice Hall, 1999.