TRABALHO DE GRADUAÇÃO

SEPARAÇÃO DE FONTE SONORA MONOFÔNICA

Daniel Bauchspiess

Brasília, janeiro de 2021

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

SEPARAÇÃO DE FONTE SONORA MONOFÔNICA

Daniel Bauchspiess

Relatório submetido como requisito parcial de obtenção
de grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Marcus Vinicius Lamar, CIC/UnB _____
Orientador

Prof. Tatiana Olivieri Catanzaro, MUS/UnB _____
Coorientadora

Brasília, janeiro de 2021

Dedicatória

Dedico este trabalho à minha família

Daniel Bauchspiess

RESUMO

Neste projeto, serão desenvolvidas redes neurais artificiais capazes de receber um sinal de áudio e classificar ou separar as fontes sonoras detectadas no mesmo. A entrada da rede pode ser dada pelos descritores do sinal sonoro, ou o sinal sonoro puro. Até o presente momento, foram desenvolvidas duas redes neurais (uma Multilayer Perceptron e uma rede de Elman) que recebem os descritores do sinal de áudio e realizam a classificação do instrumento detectado, obtendo acurácias acima de 95%.

Palavras Chave: Redes Neurais, Música, Separação de fonte sonora, MATLAB, Descritores sonoros

ABSTRACT

In this project, it will be developed artificial neural networks capable of receiving an audio signal and classifying or separating the sound sources detected. The network entries can be the descriptors of the sound signal, or the pure sound signal. Until now, two neural networks has been developed (one Multilayer Perceptron and one Elman network) that receive the audio signal descriptors and classify the detected instrument, with accuracy above 95%.

Keywords: Neural Networks, Music, Sound source separation, MATLAB, Sound descriptors

SUMÁRIO

1 Introdução 1
1.1 Contextualização 1
1.2 Definição do problema 1
1.3 Objetivos do projeto 1
1.4 Apresentação do manuscrito 2

2 Fundamentos 3
2.1 Redes neurais artificiais 3
2.1.1 Treinamento 4
2.1.2 Redes multiple-layer feedforward 5
2.1.3 Redes recorrentes 5
2.2 Acústica 6
2.2.1 Descritores sonoros 7

3 Metodologia 10
3.1 Amostras 10
3.2 Classificação baseada em descritores 11
3.2.1 Multilayer Perceptron 11
3.2.2 Rede de Elman 13

4 Resultados 15
4.1 Amostras 15
4.2 Classificação baseada em descritores 16
4.2.1 Multilayer Perceptron 16
4.2.2 Rede de Elman 16

5 Conclusões 17
5.1 Perspectivas Futuras 17

REFERÊNCIAS BIBLIOGRÁFICAS 18

6 Programas utilizados 19

LISTA DE FIGURAS

2.1 Implementação do modelo de neurônio proposto por McCulloch e Pitts[1] 4
2.2 Representação de uma rede multiple-layer feedforward[1] 5
2.3 Representação de uma rede recorrente[1] 6
2.4 Representação de uma rede de Elman[2] 7

3.1 Sinal de áudio de um contrabaixo tocado de forma ordinária, mezzo forte, na primeira corda e na nota B4. 11
3.2 Rede MLP criada na classificação por descritores. 12
3.3 Rede Elman criada na classificação por descritores. 13

4.1 Comparação entre o sinal apresentado na figura 3.1 e o mesmo após o pré-processamento. 15

LISTA DE SÍMBOLOS

Grupos Adimensionais

K	Quantidade de grupos para treino com K-Fold

Siglas

MIR	*Music Information Retrieval*
RNA	*Rede Neural Artificial*
MLP	*Multilayer Perceptron*

Capítulo 1

Introdução

1.1 Contextualização

Dentro de uma peça de música, existem muitas informações que podem ser extraídas e que seres humanos reconhecem de forma natural. Informações mais locais como a força ou o ritmo da música, ou informações mais gerais como o gênero musical, os instrumentos tocados, dentre outras. A capacidade de extrair essas informações de forma automática pode abrir muitas portas na área acadêmica e comercial, possibilitando estudos aprofundados de cada peça, seleção de músicas semelhantes, sistemas que respondem ao andamento da música e assim por diante. A área crescente que estuda as formas de extrair essas informações é a de Recuperação da Informação da Música (MIR, Music Information Retrieval). [3]

1.2 Definição do problema

Dentre os ramos de pesquisa de MIR, temos a área de identificação e separação de fonte sonora. Ao escutar música, o ser humano é capaz de identificar os instrumentos sendo tocados em poucos instantes de tempo, dependendo da familiaridade que tem com o mesmo, e até mesmo focar a escuta em algum instrumento específico. Este ramo busca resultados semelhantes, identificando os instrumentos presentes em um sinal de áudio e fornecendo novos sinais com os instrumentos isolados.

1.3 Objetivos do projeto

Este projeto é voltado ao problema da separação de fonte sonora. O objetivo é desenvolver um sistema que, a partir de um sinal de áudio onde mais de um instrumento musical está presente, seja capaz de identificar tais instrumentos e os separar em canais distintos, sendo possível recuperar com fidelidade o sinal original produzido pelo instrumento isolado.

Este sistema será uma rede neural, treinada para identificar um conjunto limitado de instru-

mentos. Serão feitas algumas abordagens, variando a arquitetura da rede neural e quais dados serão utilizados para o treinamento da rede. Em um primeiro momento, será feita apenas a identificação do instrumento em sinais de áudio com apenas um deles presente, e em seguida a rede desenvolvida deverá ser capaz de separar o sinal de cada um dos instrumentos em um sinal com ambos misturados. Mais detalhes a respeito da abordagem são tratados na Sessão 4.

1.4 Apresentação do manuscrito

Este documento é composto pelos capítulos de introdução, fundamentos, metodologia, resultados e conclusão. Na introdução, é apresentado o objetivo do projeto, com o problema a ser tratado e seu contexto. Em fundamentos, é apresentado todo o embasamento teórico necessário para que o projeto possa ser desenvolvido, passando pelas ferramentas utilizadas de redes neurais e pelos conteúdos utilizados da área de acústica. No capítulo de metodologia, são apresentados todos os sistemas feitos, a forma como foram implementados e os testes realizados, cujos resultados são apresentados no capítulo de resultados. Por fim, no capítulo de conclusões os resultados obtidos são analisados e o êxito na proposta do projeto é avaliada. Também são considerados os próximos passos para melhoria do sistema.

Capítulo 2

Fundamentos

2.1 Redes neurais artificiais

Redes Neurais Artificiais (RNA) são sistemas que se baseiam em sistemas nervosos, buscando os benefícios da aprendizagem."Redes neurais artificiais são modelos computacionais inspirados pelo sistema nervoso de seres vivos. Elas possuem a habilidade de adquirir e manter conhecimento (baseadas em informação) e podem ser definidas como um conjunto de unidades processadoras, representadas por neurônios artificiais, interligadas por muitas interconexões"[1].

O modelo mais simples de neurônio artificial, proposto por McCulloch e Pitts (1943) e apresentando as características principais de um neurônio biológico, pode ser implementado como na Figura 2.1 [1]. Nesse modelo, destaca-se os seguintes elementos:

- *Sinal de entrada do neurônio*, representado pelo conjunto $\{x_1, x_2, ..., x_n\}$.
- *Pesos associados a cada entrada*, representados pelo conjunto $\{w_1, w_2, ..., w_n\}$.
- *Agregador linear*, representado pelo símbolo de somatória $\sum$.
- *Bias*, representado pelo símbolo θ.
- *Potencial de ativação*, representado por u.
- *Função de ativação*, representado por g(.).
- *Sinal de saída*, representado por y.

Neste modelo, cada uma das entradas x_n é dado por um valor numérico. Este valor é multiplicado por seu peso correspondente w_n, e cada produto de entrada com seu peso é somado no agregador linear. Desta forma, o resultado dessa operação pode ser visto como uma soma ponderada de todas as entradas do neurônio. O resultado desta soma é subtraído do valor fornecido pelo bias, considerado um limiar de ativação. O potencial de ativação (u) é o valor numérico resultante desta operação de subtração. Por fim, este valor é utilizado como variável na função de

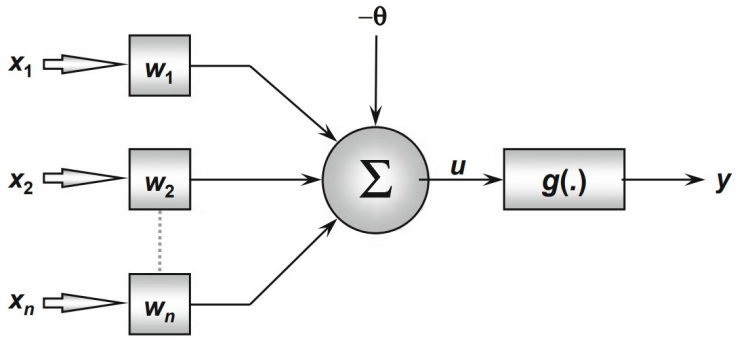

Figura 2.1: Implementação do modelo de neurônio proposto por McCulloch e Pitts[1]

ativação g(u), cujo principal objetivo é limitar a saída em uma faixa razoável de valores, e assim é produzida a saída y. Esta saída pode ser o valor final de saída do sistema, ou pode ser utilizada por outros neurônios. [1]

Uma rede neural é formada por neurônios conectados entre si e com a entrada, e estes são agrupados em camadas. Os três tipos de camadas são:

- *Camada de entrada*, responsável por receber o sinal de entrada externo. Cada sinal de entrada possui uma saída esperada, a ser comparada com a saída da rede neural.
- *Camadas intermediárias, escondidas ou invisíveis*, formada por neurônios responsáveis por extrair padrões associados ao processo sendo analisado.
- *Camada de saída*, também formada por neurônios, cujas saídas serão a saída da rede neural, resultado do processamento dos neurônios das camadas anteriores.

Na Figura 2.2, representando uma rede neural, pode-se ver as camadas destacadas, havendo uma camada de entrada com n entradas, duas camadas intermediárias, possuindo n_1 neurônios na primeira e n_2 neurônios na segunda, e uma camada de saída com m neurônios. A disposição dos neurônios, como eles estão interconectados e como as camadas da rede são compostas caracterizam as principais arquiteturas de redes neurais [1]. As arquiteturas utilizadas neste trabalho, que serão mais detalhadas nas sessões 2.1.2 e 2.1.3, são as redes multiple-layer feedforward e as redes recorrentes.

2.1.1 Treinamento

Em geral, a cada entrada aplicada na rede, é obtida uma saída que dependerá, dentre outras informações, dos valores dos pesos de entrada de cada neurônio e de seus bias. O processo de treinamento consiste em adaptar esses valores internos de forma que a saída fique o mais próximo do esperado para uma grande quantidade de dados de entrada.

Para o caso de treinamento supervisionado, que é o caso utilizado em todas as redes deste projeto, durante a etapa de treinamento, cada entrada do conjunto de dados de treinamento é previamente associada a uma saída esperada, denominado target (ou alvo, em português). Então,

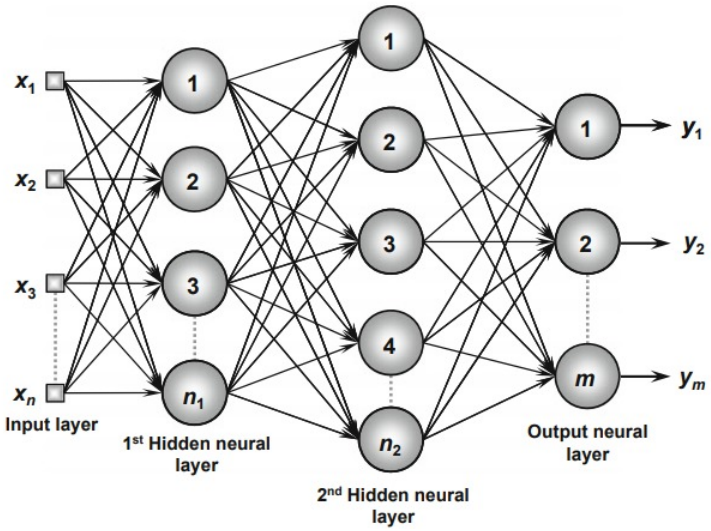

Figura 2.2: Representação de uma rede multiple-layer feedforward[1]

durante o treinamento, uma entrada é aplicada na rede, e a saída obtida é comparada com a saída esperada, e esta diferença é tomada como base para a adaptação dos pesos e bias presentes na rede. Este processo pode ser repetido algumas vezes para todos os dados do conjunto de treinamento, até que haja pouca alteração sendo feita nos parâmetros internos da rede, e espera-se que a saída da rede aproxime-se do valor de target para as entradas aplicadas seguintes.

2.1.2 Redes multiple-layer feedforward

Redes multiple-layer feedforward possuem uma ou mais camadas intermediárias (além das camadas de entrada e saída), e o fluxo da informação sempre segue uma mesma direção, da entrada até a saída [1]. Isto significa que as saídas dos neurônios de uma camada conectam-se apenas com os neurônios de camadas posteriores, mais próximas à camada de saída, nunca com a própria camada ou com camadas anteriores. Um exemplo de uma rede multiple-layer feedforward pode ser visto na Figura 2.2. Esta rede possui duas camadas intermediárias. Nota-se que todas as conexões entre camadas ocorrem seguindo da esquerda para a direita, no sentido da saída da rede.

Dentre as redes que utilizam a arquitetura multiple-layer feedforward, uma das principais é a Multilayer Perceptron (MLP), que é a rede aplicada neste trabalho. Como mencionado para redes feedforward, esta rede possui pelo menos uma camada intermediária e o fluxo de informação é unidirecional, da entrada à saída da rede. Além disso, seu treinamento é supervisionado e utiliza o método de backpropagation, e a quantidade de neurônios em cada camada é variável.

2.1.3 Redes recorrentes

Redes recorrentes são caracterizadas pelo fato de que a saída de um ou mais neurônio serve de retroalimentação para outros neurônios[1], servindo de entrada de neurônios na mesma camada ou

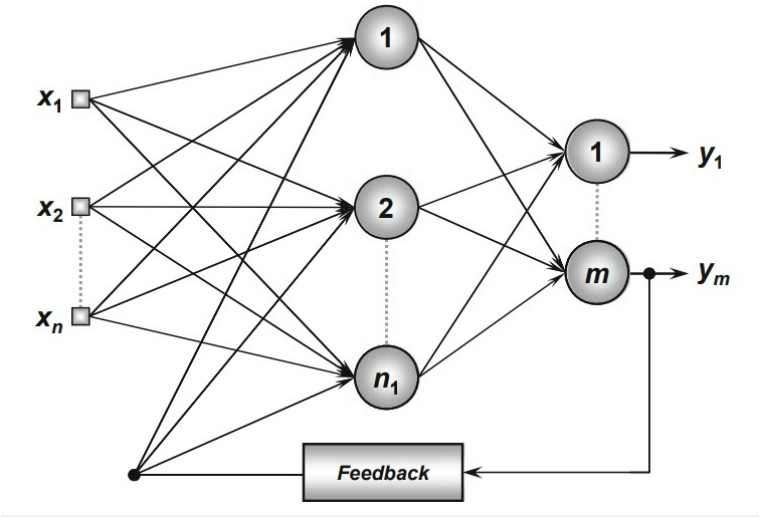

Figura 2.3: Representação de uma rede recorrente[1]

em camadas anteriores, fazendo com que o fluxo de informação não seja unidirecional. A Figura 2.3 apresenta um exemplo de uma rede neural recorrente. Nela, a saída do neurônio m da camada de saída é utilizada como entrada na camada intermediária.

Essa retroalimentação pode ser associada a uma ou mais unidades de atraso. Isso significa que o sinal que está sendo passado não será utilizado como entrada dos neurônios em conjunto com o sinal de entrada que o gerou, e sim com o sinal de entradas posteriores.

O atributo de retroalimentação permite que essas redes sejam utilizadas em sistemas variantes no tempo [1], considerando que cada classificação considerará não só o dado inserido na rede em um instante, mas também os dados anteriores a este. Um exemplo de rede desta arquitetura é a rede de Elman, utilizada neste projeto e detalhada mais adiante.

2.1.3.1 Rede de Elman

Uma rede de Elman é um tipo específico de rede recorrente. É composta pela camada de entrada, uma camada intermediária e a camada de saída. A saída da camada intermediária é retroalimentada à entrada da mesma camada, permitindo a rede reconhecer padrões temporais [4]. A Figura 2.4 apresenta uma representação de uma rede de Elman [2]

2.2 Acústica

Tendo em vista que o sinal a ser utilizado como entrada das redes neurais artificiais criadas é uma representação do som, entender suas propriedades acústicas trará benefícios para melhor modelar a rede. Dentre suas áreas de estudo, serão abordadas aqui formas de caracterização de sinais sonoros, referidas como descritores sonoros.

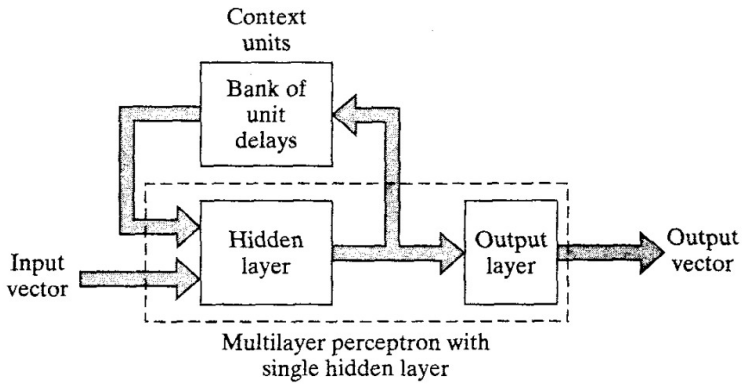

Figura 2.4: Representação de uma rede de Elman[2]

2.2.1 Descritores sonoros

Descritores sonoros, como o nome indica, são abordagens para descrever um sinal sonoro a partir de características extraídas do mesmo. Os descritores que serão abordados aqui são todos obtidos a partir da análise do espectro em frequência dos sinais.

2.2.1.1 Centroide Espectral

O Centroide Espectral, é o baricentro do espectro [5]. Ele é calculado como:

$$\mu = \frac{\sum_{i=0}^{n-1} f[i]a[i]}{\sum_{i=0}^{n-1} a[i]} \tag{2.1}$$

Onde:

- n é metade do tamanho da janela.
- i é o índice do vetor que compõe o sinal no domínio da frequência.
- $a[i]$ é a amplitude do sinal na frequência indicada pelo índice i, na parte real do cálculo da FFT (Fast Fourier Transform).
- $f[i]$ é a frequência associada ao índice i, onde:

$$f[i] = i * \frac{frequênciadeamostragem}{tamanhodajaneladaFFT}$$

- μ é o Centroide Espectral, em hertz

No MATLAB, o Centroide Espectral pode ser obtido através da função spectralCentroid(x,f) [1].

[1]Mais informações da função em https://www.mathworks.com/help/audio/ref/spectralcentroid.html

2.2.1.2 Variância Espectral

Variância Espectral é dada como a variância do Centroide Espectral [5], e pode ser calculada como:

$$\nu = \frac{\sum_{i=0}^{n-1}(f[i]-\mu)^2 a[i]}{\sum_{i=0}^{n-1} a[i]} \tag{2.2}$$

No MATLAB, a Variância Espectral pode ser obtido através da função spectralSpread(x,f) [2].

2.2.1.3 Declive Espectral

Declive Espectral é uma estimação da quantidade de magnitude espectral decrescendo [5], calculada por:

$$slope = \frac{1}{\sum_{i=0}^{n-1} a[i]} \frac{n\sum_{i=0}^{n-1} f[i]a[i] - \sum_{i=0}^{n-1} f[i]\sum_{i=0}^{n-1} a[i]}{n\sum_{i=0}^{n-1} f^2[i] - (\sum_{i=0}^{n-1} f[i])^2} \tag{2.3}$$

No MATLAB, o Declive Espectral pode ser obtido através da função spectralSlope(x,f) [3].

2.2.1.4 Decrescimento Espectral

É semelhante ao Declive Espectral, representando a quantidade de magnitude espectral decrescendo, mas em tese é mais correlacionado à percepção humana [5]. Sua fórmula é:

$$decrease = \frac{\sum_{i=0}^{n-1} a[i] - a[1]}{\sum_{i=0}^{n-1} a[i](i-1)} \tag{2.4}$$

No MATLAB, o Decrescimento Espectral pode ser obtido através da função spectralDecrease(x,f) [4].

2.2.1.5 Ponto de Roll-Off Espectral

O ponto de Roll-Off Espectral é a frequência $f_c[i]$ em que uma porcentagem do sinal fica abaixo desta frequência, por padrão sendo utilizado 95% do sinal [5]. O ponto de Roll-Off é calculado por:

$$\sum_{i=0}^{f_c[i]} a^2[f[i]] = x \sum_{i=0}^{n-1} a^2[f[i]] \tag{2.5}$$

Onde:

[2]Mais informações da função em https://www.mathworks.com/help/audio/ref/spectralspread.html
[3]Mais informações da função em https://www.mathworks.com/help/audio/ref/spectralslope.html
[4]Mais informações da função em https://www.mathworks.com/help/audio/ref/spectraldecrease.html

- $f_c[i]$ é o ponto de Roll-Off.
- x é a porcentagem de energia Roll-Off acumulada.

No MATLAB, o ponto de Roll-Off Espectral pode ser obtido através da função spectralRolloffPoint(x,f) [5].

[5] Mais informações da função em https://www.mathworks.com/help/audio/ref/spectralrolloffpoint.html

Capítulo 3

Metodologia

3.1 Amostras

Para que as redes desenvolvidas pudessem ser treinadas, foi utilizado um banco de amostras sonoras, provenientes do banco "IRCAM solo instruments", da UVI [6]. As amostras sonoras utilizadas estão no repositório deste trabalho [1], na pasta "Samples-SC". Esse banco de amostras possui diversos sinais de áudio de curta duração (na ordem de segundos), e cada sinal possuindo uma combinação única dos seguintes parâmetros:

- *Instrumento:* Cada sinal possui o som de apenas um instrumento. Neste trabalho são utilizados sinais de flauta e contrabaixo.
- *Forma de tocar:* O timbre do instrumento é alterado a depender da forma com que é tocado. Neste trabalho os sinais de áudio correspondem apenas ao sinal de áudio tocado de forma ordinária.
- *Intensidade:* O instrumento do sinal pode ser tocado com intensidades diferentes, variando entre pianíssimo (pp, intensidade fraca), mezzo forte (mf, intensidade intermediária) e fortissimo (ff, intensidade forte). Todas essas intensidades de sinal foram exploradas no trabalho.
- *Corda:* Exclusivo para instrumentos de cordas (como o cnotrabaixo utilizado), o mesmo som pode ser tocado em cordas diferentes, fator que altera o timbre do mesmo.
- *Nota:* Nota em que o instrumento é tocado, variando em todo o seu alcance.

A figura 3.1 mostra o exemplo de um desses sinais, utilizado também para treinos. O sinal é de um contrabaixo tocado de forma ordinária e mezzo forte, na primeira corda e com a nota B4.

Por meio deste gráfico, percebe-se um problema de utilizar o sinal em sua forma pura: em sua duração total, há períodos em que o instrumento não é tocado. Estes períodos não podem ser inseridos como treinamento da rede, para que a mesma não identifique silêncio como sinal de um

[1] Link para o repositório: https://github.com/minibauchspiess/Separacao-de-Fonte-Sonora

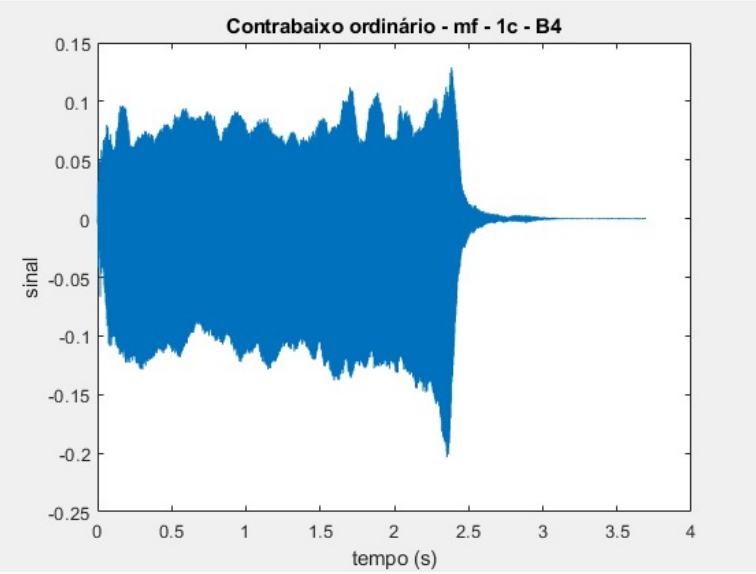

Figura 3.1: Sinal de áudio de um contrabaixo tocado de forma ordinária, mezzo forte, na primeira corda e na nota B4.

instrumento. Além disso, desejamos treinar as redes apenas nos momentos em que o sinal está estável, ou seja, nos momentos em que o mesmo se encontra na sustentação, eliminando o ataque e o decaimento. Assim, faz-se necessário realizar um pré-processamento em cada sinal de áudio, mantendo apenas os trechos correspondentes à sustentação da nota, e o utilizando por completo para o treinamento. Esse pré-processamento fora feito selecionando arbitrariamente os instantes em que o sinal supostamente estaria na sustentação.

3.2 Classificação baseada em descritores

Para a primeira etapa do trabalho, foram feitas análises das amostras sonoras a partir de cinco dos seus descritores (Centroide Espectral, Variância Espectral, Declive Espectral, Decrescimento Espectral e Ponto de Roll-Off Espectral). O objetivo é criar uma rede que receba como entrada os descritores de uma janela de tempo e forneça como saída o instrumento identificado. Ressalta-se que deseja-se apenas identificar o instrumento, e não o separar do sinal de áudio.

Nas redes criadas, a camada de entrada possui cinco neurônios, um para cada descritor, e a camada de saída possui dois neurônios, cada um associado a um dos instrumentos. Cada neurônio da camada de saída pode ter valores entre 0 a 1, representando a probabilidade de o sinal de entrada pertencer ao instrumento a que o neurônio está associado.

3.2.1 Multilayer Perceptron

A primeira rede testada foi uma Multilayer Perceptron, com apenas uma camada intermediária de 14 neurônios e sem realimentação entre camadas. No código 6.1, nas linhas 32 a 77 é definida a arquitetura da rede e seus parâmetros de treinamento. Por meio do comando "view(net)", obtém-se

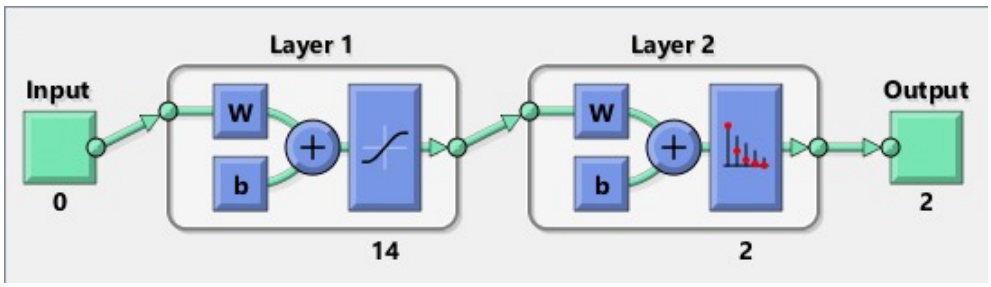

Figura 3.2: Rede MLP criada na classificação por descritores.

uma representação visual da rede, apresentada na figura 3.2.

Nessa primeira etapa não foram utilizados os sinais de áudio pré-processados. O treinamento fora realizado com dois sinais de áudio: um sinal de flauta e um sinal de contra-baixo, ambos instrumentos tocados na nota B4 (493.88 Hz), vindos do banco de amostras já apresentado. O target para contrabaixo foi o vetor [0 1], e para flauta, o vetor [1 0] (o primeiro neurônio de saída associado à flauta, o segundo associado ao contra-baixo). Cada descritor era obtido a partir de uma janela de 1024 amostras de áudio. Para manter um balanceamento entre os sinais de flauta e contra-baixo, foram utilizadas apenas 180 janelas de cada sinal, mantendo os instantes intermediários (descartando o começo e o final do sinal).

Foi utilizado o método K-Fold de validação cruzada, com K valendo 10. Dentre o conjunto separado como entrada de treinamento da rede, 20% foi utilizado para validação e 80% para treino. O conjunto de teste separado no K-Fold é utilizado por fora da função "train".

Outras informações do treinamento e arquitetura a se destacar são:

- *Método de medição da performance da rede:* Erro Quadrático Médio
- *Função de treinamento:* Levengerg-Marquardt
- *Quantidade máxima de épocas:* 10000
- *Função de transferência da camada intermediária:* tansig (tan-sigmoid)
- *Função de transferência da camada de saída:* softmax (necessário para que todas as saídas fiquem entre 0 e 1, e a somatória das mesmas seja 1)

Na etapa de preparação das amostras sonoras, cada sinal de áudio foi transformado em uma matriz 180x5, sendo cada coluna da matriz um vetor com cada um dos cinco descritores mencionados, e cada linha sendo um período de tempo comportando 1024 amostras. Essa etapa contou com o auxílio da função "GetDescriptors"e "BuildSampleMatrix". A função "GetDescriptors"tem como objetivo retornar os 5 descritores de um sinal de áudio, recebendo informações do sinal como argumento, assim como o tamanho da janela de cada conjunto de descritores e quantas janelas são utilizadas no sinal de áudio, utilizando apenas as janelas centrais. A função "BuildSampleMatrix"recebe o caminho para o sinal de áudio, a qual target ela pertence e monta a matriz

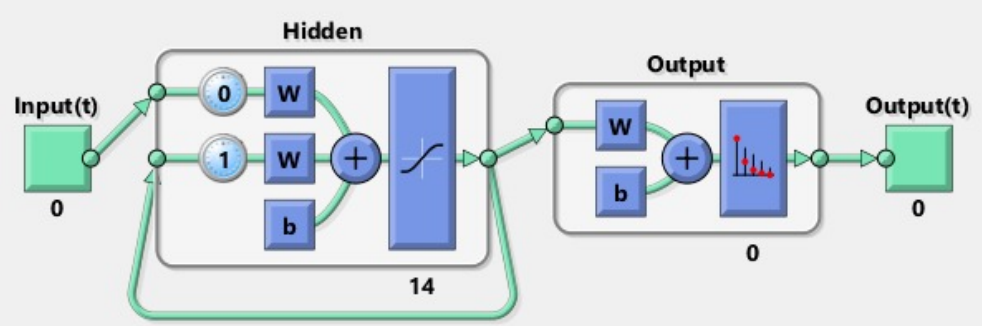

Figura 3.3: Rede Elman criada na classificação por descritores.

180x5 mencionada (com auxílio da função "GetDescriptors"), assim como uma matriz 180x2, correspondendo ao target para cada uma das linhas de entrada. Essas funções podem ser vistas no repositório, na pasta "DescriptorsPhase/ShallowNeuralNetwork".

Para a geração dos K grupos utilizados no K-Fold, foi utilizada a função "KfoldGroups". Essa função recebe todas as matrizes criadas com "BuildSampleMatrix"e monta K grupos com as linhas de ambas matrizes, mantendo o balanceamento entre as matrizes utilizadas para cada grupo. Isto é feito tanto com a entrada de descritores e seus respectivos targets, mantendo a correspondência entre uma entrada e seu target.

Na etapa de treinamento, são feitas 10 rodadas de treinamento (equivalentes a K), sendo que e cada rodada um grupo K-Fold diferente é selecionado para servir como teste, e o restante é concatenado para ser treinado pela rede. A rede é inicializada com pesos aleatórios, e então é treinada. Nota-se aqui que a rede treinada é mantida em uma variável diferente da criada como template, armazenando em código um vetor de células de redes treinadas "trainedNet", e mantendo o template "net"não treinado. É armazenado esse vetor para que cada rede treinada possa ser avaliada posteriormente.

Caso a rede treinada não tenha convergido (ela não é capaz de identificar o instrumento, possui uma acurácia muito baixa), ela é desconsiderada e uma nova rede é treinada para aquele mesmo conjunto de teste e treino. Ao final do treino de uma rede que convergira, o conjunto K-Fold de teste é alimentado como entrada da rede treinada, e sua saída é comparada com o respectivo target, e com isso tem-se uma medida da acurácia daquela rede. Ao final de todo o processo, é calculada a acurácia média e sua variância de todas as redes treinadas, assim como a quantidade de redes que não convergiram.

3.2.2 Rede de Elman

Foi criada uma rede de Elman com 14 neurônios na camada escondida e delay de 1 instante de amostragem na realimentação. A entrada permanece com 5 neurônios, um para cada descritor, e a saída com 2 neurônios, cada um associado a um dos instrumentos. Por meio do comando "view(net)", obtém-se uma representação visual da rede, apresentada na figura 3.3.

O código 6.2 foi o script principal utilizado para esta etapa. "SampleParams"é um script

auxiliar utilizado para carregar variáveis de tratamento das amostras. Nele, determina-se que a saída referente ao contrabaixo é o vetor coluna [0; 1], e a saída referente à flauta, o vetor coluna [1; 0] (o primeiro neurônio associado à flauta, o segundo associado ao contrabaixo).

Em "SampleParams"também se determina o caminho para os sinais de áudio, utilizando nessa etapa os sinais pré-processados. Cada instrumento possui sua própria pasta, e o programa realiza o treinamento da rede a partir de todos os arquivos em cada uma das pastas, ficando cada target associado à pasta daquele instrumento. Por fim, determina-se também o tamanho da janela utilizada para cálculo dos descritores como 1024, o tamanho da sequência de vetores de descritores utilizada como entrada da rede como 4, e o valor de K da validação cruzada de K-Fold como 10.

Seguindo adiante no código, temos a criação da rede de Elman por meio da função "NetParams". Nessa função é determinada a arquitetura da rede e seus parâmetros de treinamento, dentre os quais destacamos:

- Função de treinamento: Levenberg-Marquardt
- Função de transferência da camada de saída: sorfmax
- gradiente mínimo: 1e-20 (condição de parada do treinamento)

Para preparação das amostras, temos a função "FolderMatrix", cujas entradas são o caminho para a pasta onde estão localizadas os sinais de áudio de um instrumento, o target determinado para aquele instrumento e o tamanho da janela utilizada para cálculo dos descritores. Sua saída é um conjunto de matrizes de descritores e respectivos targets, sendo que cada sinal de áudio possui uma matriz correspondente. Cada vetor coluna dessa matriz é um conjunto dos cinco descritores para uma janela de tempo do sinal de áudio correspondente. A função "FolderMatrix"também utiliza a função "GetDescriptors"para recuperar os descritores de todas as janelas do sinal de áudio.

Para geração dos grupos para a validação cruzada por K-Fold, é utilizada a função "BuildK-Groups". Esta recebe todas as matrizes geradas por "FolderMatrix", a quantidade de grupos que deve ser criada e o tamanho da sequência de vetores de uma matriz que é considerada uma entrada da rede. Sua saída são dois vetores de células, um vetor com os grupos de entrada da rede, e o outro com seus respectivos targets. No vetor de entrada, cada elemento é um dos grupos K-Fold, e dentro desse grupo há sequências de vetores de descritores de ambos instrumentos, extraídos das matrizes criadas com "GetDescriptors"e unidos por meio da função "catsamples". Dentro dos grupos há um balanceamento entre sequências vindas de cada um dos instrumentos.

Foram feitas dez rodadas de treinamento, a cada momento isolando um dos grupos obtidos com "BuildKGroups"para teste, e unindo os demais grupos para treinar a rede de Elman. Cada rede treinada é armazenada em um vetor de redes, para análises futuras. Ao final de cada treinamento, é inserido o conjunto de testes na rede treinada, e a saída obtida é comparada com o respectivo target, obtendo, assim, uma medida da acurácia da rede.

Capítulo 4

Resultados

4.1 Amostras

A figura 4.1 mostra o mesmo sinal da figura 3.1 em azul, e o trecho do mesmo que fora isolado após o pré-processamento, em laranja, para ser utilizado no treinamento. Os sinais pré-processados foram inseridos em uma pasta separada.

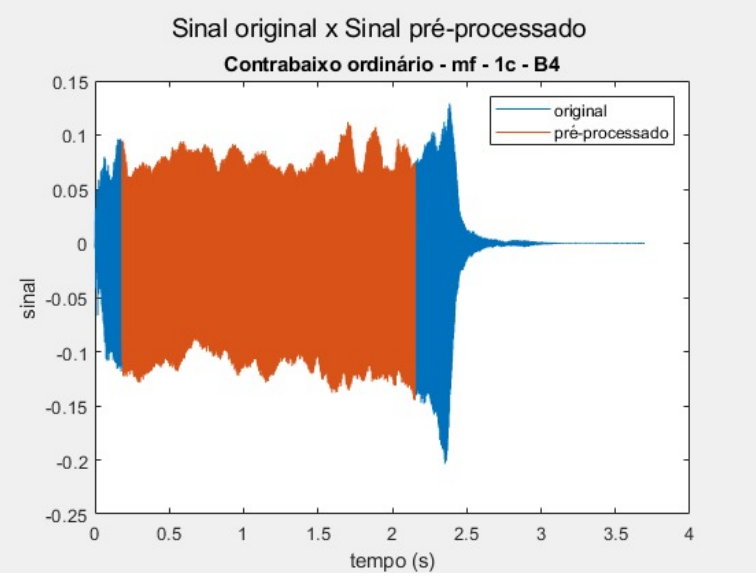

Figura 4.1: Comparação entre o sinal apresentado na figura 3.1 e o mesmo após o pré-processamento.

4.2 Classificação baseada em descritores

4.2.1 Multilayer Perceptron

Após os treinamentos da rede descrita na sessão 3.2.1, foi obtida uma acurácia média de 97.78%, variância de $6.5158 * 10^{-6}$%, e todas as redes treinadas convergiram.

4.2.2 Rede de Elman

Após os treinamentos da rede descrita na sessão 3.2.2, foi obtida uma acurácia média de 98.75% e variância de 0.0357%.

Capítulo 5

Conclusões

A partir dos resultados obtidos, pode-se afirmar que as redes desenvolvidas obtiveram êxito em classificar os sinais de entrada, possuindo ambas uma acurácia acima de 95%. Entretanto, os testes foram feitos com condições muito favoráveis à distinção, apresentando apenas uma nota de cada uma, sendo que a rede tenha aprendido a reconhecer apenas aquela nota naquele timbre. Para uma rede mais robusta, é necessário que haja o treinamento com mais notas diferentes.

5.1 Perspectivas Futuras

Para continuidade do trabalho, espera-se aumentar a robustez das redes com o treinamento com mais de uma nota, conferindo a capacidade de generalização da mesma, sendo possível classificar o instrumento identificado com mais de uma nota.

Também espera-se criar mais redes neurais recorrentes utilizando como entrada o sinal de áudio puro, e não mais com os descritores de janelas de tempo. Além disso, os testes até agora apenas tentaram classificar o sinal. Utilizando o sinal de áudio como entrada, espera-se poder combinar instrumentos diferentes em um mesmo sinal de entrada, e treinar uma rede para que em sua saída seja separado as componentes do sinal de cada instrumento, realizando a separação da fonte sonora.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] SILVA, I. Nunes da. *Artificial Neural Networks*. [S.l.]: Springer International Publishing Switzerland, 2017.

[2] HAYKIN, S. *Neural Networks - A Comprehensive Foundation*. [S.l.]: Pearson Education, Inc, 1999.

[3] SANTINI, R. M.; SOUZA, R. F. d. Recuperação da informação de música e a ciência da informação: tendências e desafios de pesquisa. 2012.

[4] Srinivasan, V.; Eswaran, C.; Sriraam, N. Approximate entropy-based epileptic eeg detection using artificial neural networks. *IEEE Transactions on Information Technology in Biomedicine*, v. 11, n. 3, p. 288–295, 2007.

[5] MALT, M.; JOURDAN, E. Zsa. descriptors: a library for real-time descriptors analysis. In: . [S.l.: s.n.], 2008.

[6] UVI. *IRCAM solo instruments 2*. Disponível em: <https://www.uvi.net/en/orchestral/ircam-solo-instruments-2.htmlspecs>.

Capítulo 6

Programas utilizados

Listing 6.1: Script principal utilizado para o treinamento da rede MLP Fonte: Daniel Bauchspiess

```
%SourceClassifier - Classificador de fonte

%O programa abaixo aborda o problema da identificacao do instrumento
%musical em dois sinais de audio distintos. Para tanto, cada sinal de audio
%eh separado em amostras de tamanhos iguais, e de cada amostra sao extraidos
%5 descritores do som (spectral centroid, spread (espalhamento), slope, decrease e
%rolloff). Esses descritores serao utilizados para se determinar qual o
%instrumento tocado, utilizando-os como entrada numa rede neural rasa nao-recorrente.
%A fim de analisar o desempenho da rede de forma mais aceita
%estatisticamente, sera utilizado o metodo do k-fold, buscando qual a
%acuracia m dia das k redes treinadas. Redes que nao convergiram nao sao
%consideradas para este proposito.



%***Parametros para pegar os dados das amostras***

outCb = [0 1]; %Saada da rede correspondente a um contrabaixo
outFl = [1 0]; %Saada da rede correspondente a uma flauta
folder = 'AmostrasSonoras/Samples-SC'; %Pasta onde estao armazenadas as amostras sonoras
files = {'Cb-ord-pp-1c- B4.aif' 'Fl-ord-B4-pp.aif'};    %Arquivos a serem utilizados na entrada da rede
targets = {outCb outFl};    %Qual a sa da desejada da rede, considerando a ordem dos arquivos em "files"

winPerAudio = 180; %Quantas amostras serao extra das de cada  udio
winSize = 1024;     %Qual o tamanho de cada amostra extraido do sinal

```

```
28 k = 10;      %Quantidades de grupos do k-fold
29
30
31
32 %***Parametros da rede***
33
34 numIn = 1;              %Apenas um input, sendo este um vetor
35 numLay = 2;             %Duas camadas, sendo uma intermediaria e outra de
       output
36 biasCon = [1; 1];       %Todas camadas com bias
37 inCon = [1; 0];         %Input se conecta apenas a camada intermediaria
38 layCon = [0 0; 1 0];    %Layer intermediaria nao recebe entrada de si mesma e
       da ultima
39                         %Ultima layer recebe entrada da intermediaria e nao
                               recebe de si mesma
40 outCon = [0 1];         %Apenas ultima camada eh conectada ao output
41
42 net = network(numIn, numLay, biasCon, inCon, layCon, outCon);   %Arquitetura
       da rede criada
43
44
45
46 %***Parametros para treinamento da rede***
47
48 net.adaptFcn = 'adaptwb';
49 net.divideFcn = 'dividerand'; %Divide-se o conjunto de treinamento
       aleatoriamente
50 net.divideParam.trainRatio = 0.8;   %Divis o dos conjuntos de treinamento,
       valida   o e teste
51 net.divideParam.valRatio = 0.2;
52 net.divideParam.testRatio = 0;  %O conjunto de teste    separado e testado
       manualmente, com o m todo de k-fold
53
54 net.performFcn = 'mse';      %Mean Square Error, medida de performance da rede
55 net.trainFcn = 'trainlm';    %Fun   o de treino segue o modelo de Levenberg-
       Marquardt backpropagation
56 net.trainParam.epochs = 10000;
57 net.trainParam.min_grad = 1e-15;
58 net.trainParam.max_fail = 2500;
59 net.trainParam.mu_max = 1e99;
60
61 net.plotFcns = {'plotperform','plotconfusion'};
62
63
64
65 %***Arquitetura das camadas da rede***
66
```

```
67 %Layer 1
68 net.layers{1}.name = 'Layer 1';
69 net.layers{1}.dimensions = 14;  %14 neur nios na camada intermedi ria
70 net.layers{1}.initFcn = 'initnw';
71 net.layers{1}.transferFcn = 'tansig';
72
73 %set Layer2
74 net.layers{2}.name = 'Layer 2';
75 net.layers{2}.dimensions = 2;   %2 neur nios na camada de sa da (cada um
       indica a um dos instrumentos)
76 net.layers{2}.initFcn = 'initnw';
77 net.layers{2}.transferFcn = 'softmax';
78
79
80
81 %***Prepara  o das amostras por k-fold***
82
83 for i=1:size(files,2)
84     %Extraindo os descritores de um sinal de  udio  e inserindo-os em uma
85     %matriz de entrada
86     path = fullfile(folder, files{i});
87     [X{i}, T{i}, fs] = BuildSampleMatrix(path, targets{i}, winSize,
           winPerAudio);
88 end
89
90 %Gerando grupos para k-fold
91 [kX, kT] = KfoldGroups(X, T, k);
92
93
94 %***Treinamento dos k grupos de amostras***
95
96 discartedNetCount = 0; %Contador para as redes nao convergidas e descartadas
97 for i = 1:k
98     %Separa os k grupos de treino
99     xTrain{i} = [];
100    xTarget{i} = [];
101    for j = 1:k
102        if i ~= j
103            xTrain{i} = [xTrain{i}; kX{j}];
104            xTarget{i} = [xTarget{i}; kT{j}];
105        end
106    end
107
108    %Separa o grupo de testes
109    xTest{i} = kX{i};
110    xTestTarget{i} = kT{i};
111
```

```
112     %Reinicia a rede
113     net = init(net);
114
115     %Treina a rede, utilizando o grupo em quest o
116     [trainedNet{i}, tr{i}] = train(net, xTrain{i}', xTarget{i}');
117
118     %Caso a rede n o tenha convergido, descarta resultado e treina outra
119     while tr{i}.perf(end) > 0.15
120         discartedNetCount = discartedNetCount + 1;
121
122         net = init(net);
123         [trainedNet{i}, tr{i}] = train(net, xTrain{i}', xTarget{i}');
124     end
125
126
127     %Utilizando o vetor de teste, para testar a acur cia da rede
128     y{i} = trainedNet{i}(xTest{i}');
129
130     %Gera matriz de confusao para este teste
131     [conf{i}, confMat{i}] = confusion(xTestTarget{i}', y{i});
132 end
133
134 %Calcula acuracia media das redes treinadas
135 meanAccuracy = mean(1 - cell2mat(conf))
136 varAccuracy = var(1 - cell2mat(conf))
```

Listing 6.2: Script principal utilizado para o treinamento da rede Elman, com descritores Fonte: Daniel Bauchspiess

```
1
2 %***Carrega os parametros utilizados para obter as amostras***
3 SampleParams;
4
5 %***Cria a rede a ser utilizada***
6 net = NetParams();
7
8
9 %Geracao das amostras, com K groups
10 [CbMats, cbTargets] = FolderMatrix(fullfile(folder,cbFolder), outCb, winSize)
       ;
11 [FlMats, flTargets] = FolderMatrix(fullfile(folder,flFolder), outFl, winSize)
       ;
12
13 [X, T] = BuildKGroups({CbMats,FlMats}, {cbTargets,flTargets}, k, seqSize);
14
15
16
17 %Treina as k redes, com os grupos de teste e treino definidos
```

```
18 for i=1:k
19     %Monta as entradas de treino e teste
20     [Xtrain{i},Ttrain{i},Xtest{i},Ttest{i}] = TrainTestKGroups(X,T,i);
21
22     %Inicializa a rede e a treina
23     net = init(net);
24     trainedNet{i} = train(net,Xtrain{i},Ttrain{i});
25
26     %Testa a rede rec m treinada, adquirindo sua performance e erro ao
27     %longo do tempo
28     Y{i} = trainedNet{i}(Xtest{i});
29     [conf{i}, confMat{i}] = confusion(cell2mat(Ttest{i}), cell2mat(Y{i}));
30 end
```