5
Feature Denoising

Os capítulos 3 e 4 mostraram duas abordagens já conhecidas na literatura para a questão de ruído: extração robusta de atributos (como o SSCH e o PNCC) e wavelet denoising. Neste capítulo 5, é proposta uma terceira abordagem. Em vez de filtrar as informações indesejáveis **antes** da extração de atributos, pode-se filtrá-las **depois** da extração. Isto é, os atributos de um sinal corrompido são modificados para ficarem mais parecidos com os de um sinal limpo, como mostra a Figura 49. Por isso, o processo foi batizado de *feature denoising*.

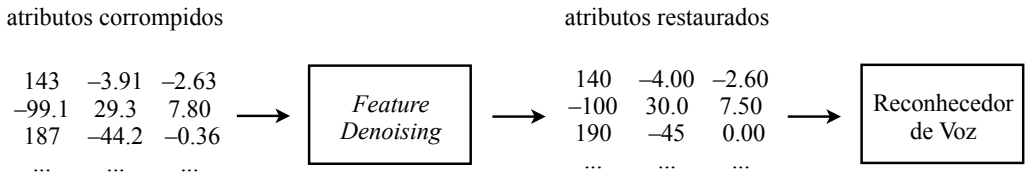

Figura 49:Restauração dos atributos extraídos, antes do reconhecimento de voz.

A grande vantagem dessa nova limpeza, comparada com a do capítulo anterior, é melhorar a informação fornecida diretamente para o reconhecedor de voz. A eliminação de valores da transformada *wavelet*, explicada nas seções 4.3 e 4.4, pode acabar removendo também dados importantes para o reconhecimento, pois não se leva em consideração a extração de atributos. Já o *feature denoising* modifica os próprios atributos, para serem mais similares aos usados no treinamento dos HMMs.

Logo, era necessário um modelo novo para transformar atributos corrompidos em atributos limpos. Na seção a seguir, é introduzido o conceito de rede neural artificial, que se mostrou muito capaz para essa transformação.

5.1
Rede Neural Artificial

Em muitas aplicações de engenharia, procuram-se meios de explicar ou prever um determinado fenômeno. Para isso, dados são coletados e usados no

ajuste de um modelo matemático, num processo chamado de treinamento supervisionado [31]. A Figura 50 traz um exemplo hipotético e simples: classificação de animais em classes, a partir de suas características. Inicialmente, o modelo recebe exemplos de animais com classificação conhecida, numa etapa de treinamento. Após o aprendizado, são apresentados animais inéditos e o modelo terá que responder corretamente quais são as suas classes.

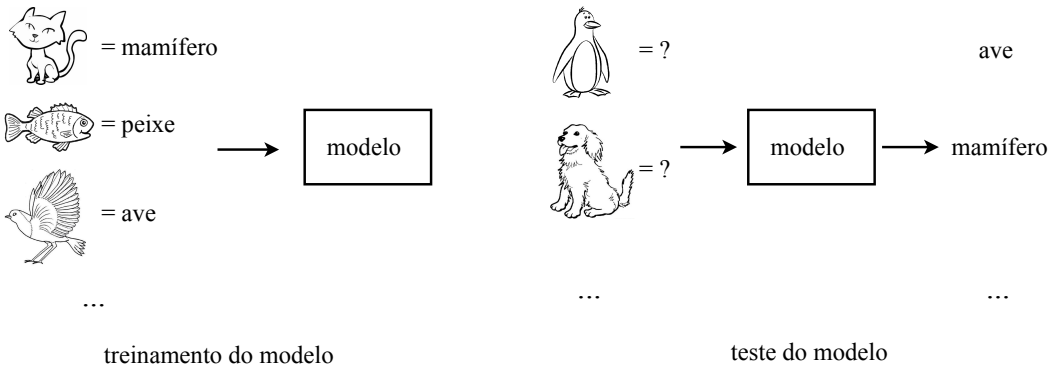

Figura 50:Exemplo da criação de um modelo matemático para classificar animais.

O modelo em questão poderia ser uma simples equação, cujos parâmetros são ajustados para associar as entradas com as saídas desejadas. Entretanto, alguns fenômenos são complexos demais para serem descritos assim. Para representá-los, existem diversos métodos alternativos. Um deles, chamado de rede neural artificial [6], foi inspirado numa característica dos seres vivos: o sistema nervoso.

A velocidade de processamento do cérebro, por incrível que pareça, é bem mais lenta que a de circuitos digitais modernos. No entanto, ele apresenta uma capacidade muito maior de processamento em paralelo e de reconhecimento de padrões – vantagens bastante úteis para a remoção do ruído.

Um cérebro biológico é formado por células simples chamadas de neurônios (ver Figura 51), que possuem alguns terminais de entrada (dendritos) e um terminal de saída (axônio). Quando a quantidade de estímulos nas entradas ultrapassa um determinado potencial elétrico, o neurônio ativa sua saída transmitindo um sinal sináptico, o qual poderá estimular ou não outros neurônios. Caso o limiar não seja ultrapassado, não ocorrerá a transmissão.

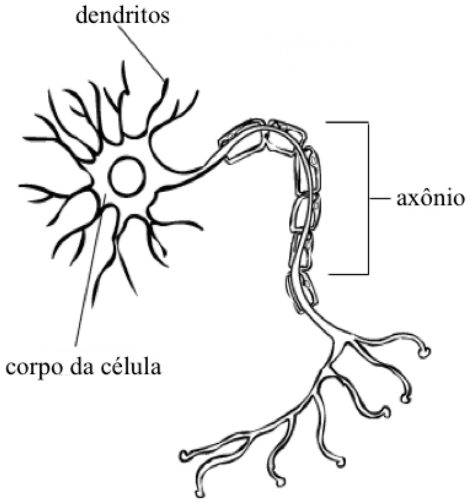

Figura 51:Esquema de um neurônio biológico.

Um único neurônio tem um comportamento bem simples, mas a conexão de bilhões deles gera uma rede complexa, que consegue controlar o corpo, armazenar memórias e raciocinar. Buscando uma capacidade similar, o neurônio artificial da Figura 52 foi criado para imitar o comportamento da Figura 51.

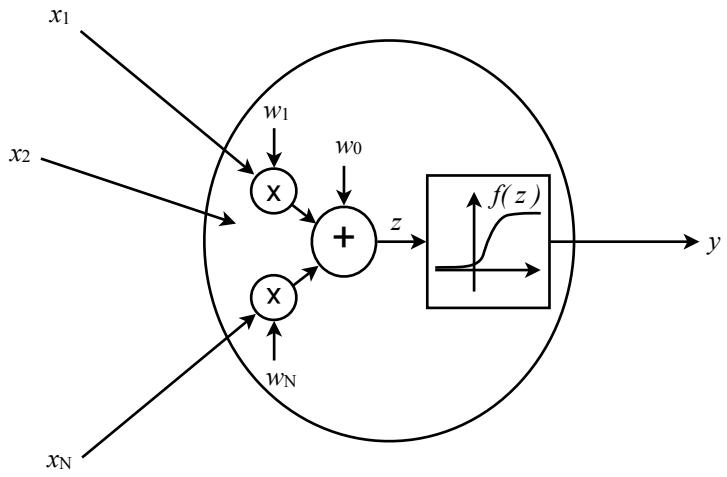

Figura 52:Esquema de um neurônio artificial.

O neurônio artificial recebe N entradas numéricas $\{x_1, x_2, \ldots, x_N\}$, as quais são ponderadas por pesos $\{w_1, w_2, \ldots, w_N\}$ e somadas através da expressão

$$z = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_N x_N \tag{41}$$

A variável z equivale então ao total de estímulos nos dendritos. Esse resultado passa por uma função de ativação $f(z)$, gerando uma saída y dada por

$$y = f(z) \tag{42}$$

Para imitar a transmissão de um neurônio biológico, a função $f(z)$ deve ter um comportamento crescente semelhante ao da Figura 52. Dois exemplos bastante usados são

$$y = \text{logsig}(z) = \frac{1}{1+e^{-z}} \tag{43}$$

$$y = \text{tansig}(z) = \frac{2}{1+e^{-2z}} - 1 \tag{44}$$

Assim como o biológico, o neurônio artificial por si só não é de grande utilidade, mas a conexão de vários deles é capaz de representar comportamentos matemáticos complexos, incluindo os não-lineares. Um modo comum de interligar os neurônios é com a configuração de camadas da Figura 53, chamada de rede neural *feedforward*: a saída de cada neurônio é propagada como uma entrada nos neurônios da camada seguinte.

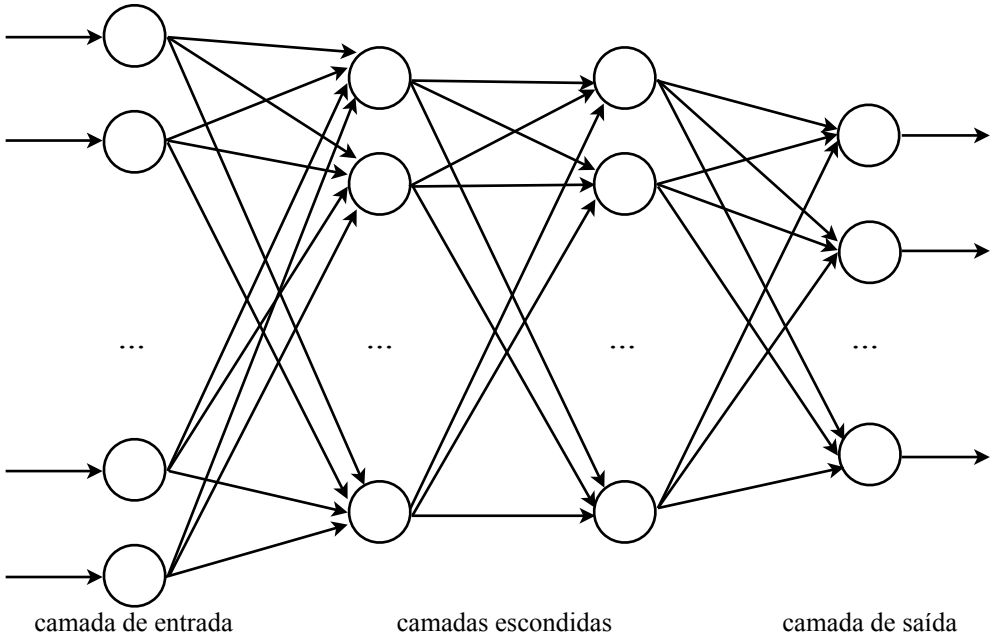

Figura 53:Rede neural *feedforward*, formada por conexões entre neurônios artificiais.

O funcionamento da rede é bem simples. A primeira camada apenas recebe as entradas numéricas do modelo e as distribui para os neurônios da camada seguintes. Esses neurônios aplicam as expressões (41) e (42) e propagam os resultados para os neurônios seguintes. Os cálculos e transmissões prosseguem até a última camada, que retorna as saídas finais do modelo.

Em [32], foi demonstrado que a configuração *feedforward* é uma aproximação universal de funções. Isso significa que existe uma rede capaz de

transformar um conjunto de entradas em quaisquer saídas desejadas. No caso deste trabalho, as entradas estão relacionadas aos atributos corrompidos por ruído; as saídas, aos atributos limpos.

Logo, para criar o comportamento de *denoising* sugerido no começo do capítulo, basta ajustar devidamente os pesos w_i e a quantidade de neurônios. Os detalhes de como isso é feito estão descritos na seção 5.2.

5.2
Configuração da Rede Neural

O primeiro passo para configurar uma rede neural é estabelecer exatamente que entradas e saídas ela terá. Para o *feature denoising* proposto, uma possibilidade seria apresentar todos os atributos do sinal na entrada e capturar todos os atributos limpos na saída. Porém isso complicaria muito o treinamento da rede, pois seria necessário ajustar o peso w de milhares de conexões. Outro problema é que o número de neurônios precisa ser fixo, enquanto o total de atributos varia de acordo com a duração do sinal.

Para resolver esses dois problemas, pode-se levar em consideração o modo como se extraem os atributos. Como visto no capítulo 3, uma quantidade fixa de valores é obtida em cada quadro do sinal, gerando diversos vetores. Portanto, uma segunda possibilidade seria fornecer à rede somente o vetor de atributos de cada quadro, um a um, separadamente, como mostra a Figura 54. Os vetores produzidos na saída seriam reunidos novamente no final e passados para o reconhecedor de voz.

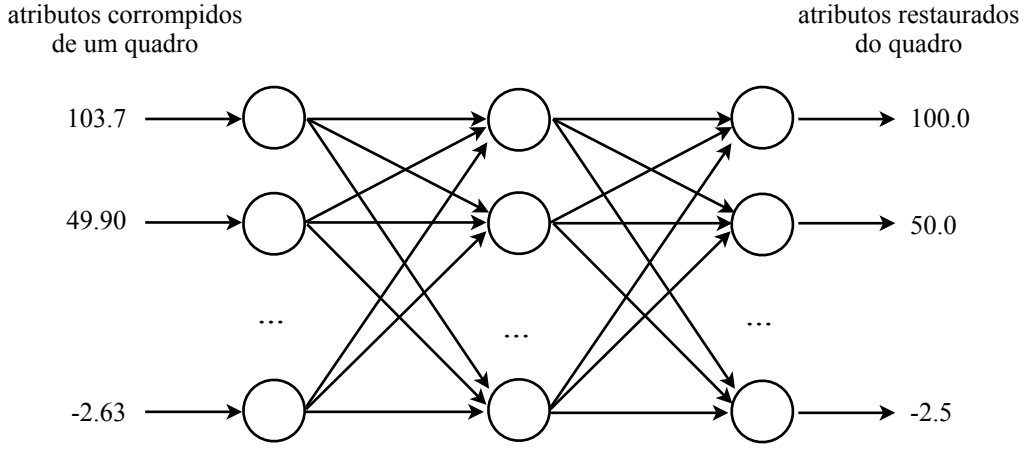

Figura 54:Configuração da rede neural com todos os atributos restaurados do quadro na saída.

Mesmo com apenas um único vetor na entrada e outro na saída, a rede ainda precisaria restaurar muitos atributos de uma só vez. Por isso, é proposto reduzir essa complexidade dividindo o problema em várias redes neurais, como mostra a Figura 55. Cada atributo do quadro é calculado na saída uma rede distinta, mas cada rede recebe todos os atributos do quadro na entrada. A ideia dessa abordagem é dividir o cálculo em redes mais simples, mas sempre aproveitando todas as informações disponíveis do quadro corrompido.

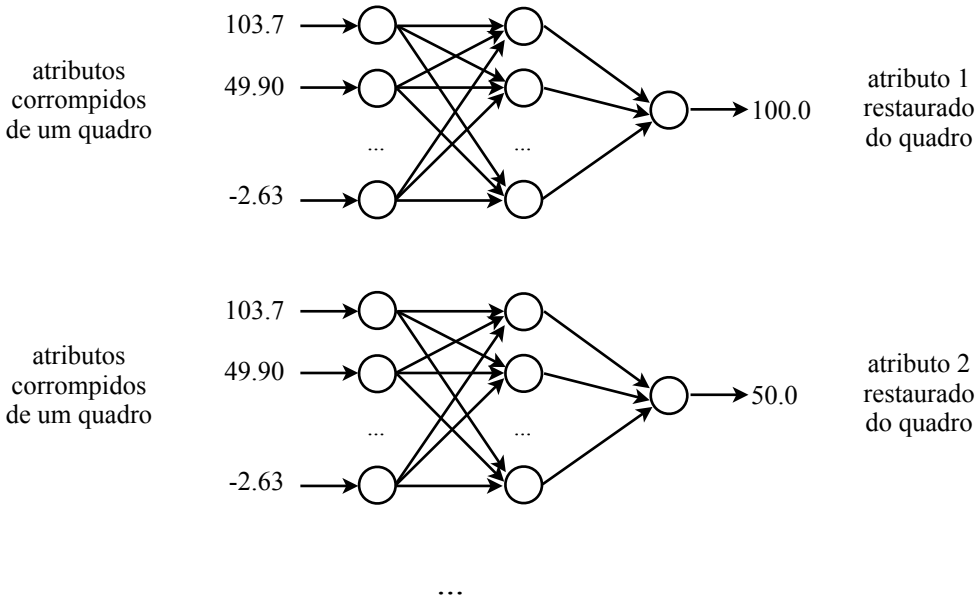

Figura 55:Configuração de várias redes neurais, cada uma com um único atributo na saída.

Após estabelecer a configuração das redes, basta treiná-las para transformar valores corrompidos em valores restaurados, usando o seguinte treinamento supervisionado:

- Alguns sinais limpos de voz são selecionados e seus atributos são extraídos
- Adiciona-se ruído a esses sinais, para corrompê-los. Os atributos dos resultados são calculados.
- Os atributos corrompidos são apresentados na entrada das redes, que ajustam seus pesos w_i para que a saída seja mais parecida com o atributo limpo respectivo.

A matemática detalhada do ajuste de pesos se encontra em [6]. Outros detalhes de configuração da rede, tais como o número de neurônios na camada escondida e a escolha da função de ativação, serão discutidos na subseção 6.3.4.