Universidade Federal Rural de Pernambuco

Departamento de Estatística e Informática

Uma Comparação Experimental de Pares de Funções de Erro e Funções de Ativação de Redes Neurais Aplicada ao Reconhecimento de Imagens

Emeson José Santana Pereira

Recife

Julho de 2015

Emeson José Santana Pereira

Uma Comparação Experimental de Pares de Funções de Erro e Funções de Ativação de Redes Neurais Aplicada ao Reconhecimento de Imagens

Orientador: Rodrigo Gabriel Ferreira Soares

Monografia apresentada ao Curso Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Recife

Julho de 2015

Ao
meu amor Kassia,
À minha família,
Ao meu orientador,
À equipe Citrus Tecnologia,
À equipe SWQuality
e a todos os amigos.

Agradecimentos

Ao meu amor Kassia, pelo constante apoio, incentivo, compreensão e paciência.

À minha família, pelo apoio e carinho dedicados a mim por todos esses anos.

Ao meu orientador, por me conduzir ao caminho correto sempre buscando extrair o melhor de mim, pela compreensão nos momento difíceis e pela dedicação a este trabalho.

Aos meus amigos Erick, Flaviano e Thays pelo apoio e por serem pessoas com quem eu posso contar.

À Ana, pela orientação profissional e pessoal passada.

À SWQuality, pelas experiências compartilhadas e pelo conhecimento adquirido ao longo dos anos que foram de grande importância para minha formação.

Ao professor Tiago, por toda experiência passada duranta minha iniciação científica que foram fudamentais para o meu crescimento profissional.

Por fim, à todos os meus amigos.

Resumo

Devido a sua não-linearidade, sua adaptabilidade e a sua grande capacidade de aprendizagem, as redes neurais artificiais (RNAs) são amplamente aplicadas ao reconhecimento de imagens. Em RNAs, além da escolha da arquitetura, outro fator importante é a escolha do par de funções de erro e ativação, visto que estão diretamente ligadas a aprendizagem de redes neurais baseada na minimização de erro.

Este trabalho tem o objetivo de analisar o impacto da escolha dos pares de função de erro e função de ativação sobre a capacidade de generalização de RNAs quando aplicadas a problemas de reconhecimento de imagens. Neste trabalho, realizamos experimentos. Comparamos os resultados estatisticamente através do método de teste de hipótese teste t de Student e analisamos os resultados obtidos.

Mostramos que o par de função de ativação linear e função de erro soma dos quadrados do erro teve, estatisticamente, o melhor resultado. Os pares de função de ativação sigmóide logística e função de erro entropia cruzada para duas classes, função de ativação *softmax* e função de erro entropia cruzada multiclasses, mostraram o segundo e o terceiro melhor desempenho preditivo, repectivamente.

Palavras-chave: redes neurais artificiais. reconhecimento de imagens. funções de erro. função ativação. experimentos. teste de hipótese.

Abstract

The artificial neural networks (ANNs) are widely applied to image recognition because their non-linearity, adaptability and learning capacity. In ANNs, beyond the architecture choice has one main factor. There is another important factor is the choice of the pair of error function and activation function, because they are directly connected to learning of neural networks based on error minimization.

The objective of this study was analyze the impact of the choice of pairs of error function and activation function on ANNs generalization capability when applied to image recognition problems. In this work, we performed experiments. We compared statistically the results based on the hypothesis testing method Student's t-test and we analyze the results.

We show that the pair of linear activation function and sum of square error function had the best result. The pairs of logistic sigmoid activation function and cross-entropy error function for two classes, softmax activation function and multiclass cross-entropy error function, showed the second and third best predictive performance, respectively.

Keywords: artificial neural networks. image recognition. error function. activation function. experiments. hypothesis testing.

Sumário

Lista de ilustrações

Lista de tabelas

1 Introdução

O cérebro humano é considerado o mais fascinante processador existente, baseado nisso foram inspiradas as redes neurais artificais (RNAs). As RNAs são modelos computacionais inspirados no cérebro humano capazes de aprender e realizar tarefas como o reconhecimento de padrões [Bishop et al., 1995, de Pádua Braga et al., 2000, Haykin, 2001]. Em reconhecimento de imagens, as redes neurais são frequentemente utilizadas com o objetivo de identificar padrões de objetos em imagens [Gonzalez and Woods, 2000].

As RNAs são formadas por neurônios artificais que têm suas saídas baseadas em funções de ativação que têm como objetivo definir a saída de um neurônio. É através da forma que os neurônios de uma rede neural estão ligados entre si que permite a RNA aprender e realizar tarefas como o renhecimento de imagens.

Na fase de aprendizagem, as RNAs passam por um processo de ajuste de parâmetros. Para este processo é comumente utilizada a aprendizagem supervisionada [de Pádua Braga et al., 2000]. Na aprendizagem supervisionada um fator importante é a função de erro que permite identificar o quão distante do esperado está a reposta da RNA.

Baseado na importância e na existência de várias funções de erro e funções de ativação, abordamos neste trabalho uma comparação experimental de pares de funções de erro e ativação aplicada ao reconhecimento de imagens.

1.1 Motivação e Contribuições

As RNAs são muito utilizadas para tarefas que envolvem reconhecimento de imagens devido a sua não-linearidade e sua capacidade inata de adaptação. No entanto, um fator importante para o sucesso da aplicação de redes neurais está em sua aprendizagem. Muitos algoritmos utilizados para treinamento de RNAs são intimamente correlacionados às funções de ativação utilizadas na rede neural e às funções de erro.

Na literatura, existe uma carência de trabalhos que estudam o impacto de pares de funções de erro e funções de ativação na capacidade de generalização das RNAs. Desta forma esperamos conceder as seguintes contribuições cientítificas:

- Uma comparação experimental, aplicada a problemas de reconhecimento de imagens.
- Uma análise sobre o impacto da escolha dos pares de funções de erro e ativação sobre a capacidade de generalização de redes neurais quando aplicadas a problemas de reconhecimento de imagens.

1.2 Objetivos

Este trabalho tem como principais objetivos:

- Realizar experimentos com bases de dados de reconhecimento de imagens para apontar o impacto de pares naturais de funções de erro e funções de ativação [Bishop et al., 1995, Dunne and Campbell, 1997] em redes neurais, utilizando nos experimentos o método de validação cruzada *k-fold*;
- Comparar estatisticamente a capacidade de generalização dos classificadores através do método de teste de hipótese *T-Test* [Fisher, 1925];
- Analisar os resultados obtidos.

1.3 Organização do trabalho

Este trabalho está organizado em oito capítulos dos quais:

- No Capítulo 1 encontra-se a introdução deste trabalho;
- Apresentamos alguns trabalhos que aplicam redes neurais em reconhecimento de imagens e estudos que abordam uma comparação de pares de funções de erro e funções de ativação, no Cápítulo 2;
- No Capítulo 3 discutimos como são representadas as imagens digitais e os passos para o processamento de imagens;
- O Capítulo 4 é destinado às redes neurais e suas arquiteturas, redes de funções de base radial e algoritmos de aprendizagem;

- Dedicamos o Capítulo 5 à definição de pares naturais de funções de erro e funções de ativação e mostramos os pares naturais e não naturais utilizados neste trabalho;
- No Capítulo 6 descrevemos as bases de dados utilizadas, os passos utilizados para realizar os ajustes de parâmetros das redes neurais e os resultados obtidos;
- O Capítulo 7 destina-se a uma análise dos resultados;
- Por fim, no Capítulo 8 descrevemos as conclusões e os trabalhos futuros.

2 Trabalhos Relacionados

Desde o surgimento do algoritmo *back-propagation* [Rumelhart et al., 1988], em meados da década de 1980, são comuns estudos envolvendo redes neurais artificiais em pesquisas voltadas a aprendizagem de máquina.

Neste capítulo, apresentaremos trabalhos com redes neurais e estudos que abordam comparações de pares de funções de erro e ativação.

2.1 Redes Neurais Aplicadas ao Reconhecimento de Imagens

Contendo uma grande base de dados, com cerca de 9298 numerais digitalizados a partir de códigos postais manuscritos de envelopes de mensagens e 3349 dígitos impressos, Le Cun et al. [1990] apresenta uma aplicação de redes neurais multicamadas em reconhecimento de códigos postais escritos à mão.

Após a etapa de pré-processamento, as imagens têm seu tamanho normalizado em 16×16 *pixels* e os valores de cada *pixel* em nível de cinza são normalizados em $[-1, 1]$.

A rede neural utilizada para classificar os padrões tem uma arquitetura bi-pirâmide com 5 camadas escondidas e 10 neurônios na camada de saída, um neurônio para cada classe, com saída em $[-1, 1]$. Quando um padrão pertencente a classe i é apresentado, a saída desejada é 1 para a unidade de saída i, e -1 para as outras unidades de saída. Seu treinamento é feito através do algoritmo *back-propagation* baseado no método do gradiente descendente.

O artigo mostra que foi possível obter um erro de treinamento de $1,1\%$ em 7291 dígitos escritos à mão e 2549 dígitos impressos, e um erro de teste de $3,4\%$ em 2007 dígitos escritos à mão e 700 dígitos impressos.

Rowley et al. [1998] apresenta uma técnica para detecção de faces em imagens em escala de cinza baseada em rede neurais de dois estágios. No primeiro estágio, regiões (janelas) de 20×20 *pixels* são obtidas da imagem para cada posição de *pixel* a fim de detectar as faces em qualquer lugar da imagem, ao mesmo passo que a imagem é reduzida em escala repetidas vezes para obter faces maiores que a janela. Em seguida, é aplicado um pré-processamento a cada janela da imagem para tentar corrigir a luminosidade e equalizar

os valores de intensidade de brilho.

Após o pré-processamento, as janelas passam através da rede neural que recebe três tipos de entradas, direcionadas a três diferentes conjuntos de unidades (neurônios) escondidas. Quatro unidades escondidas recebem 10×10 sub-regiões de *pixels*, dezesseis recebem 5×5 sub-regiões de *pixels* e seis recebem sobreposições de listras horizontais de 20×5 *pixels*. Cada uma dessas três entradas foram escolhidas com o intuito de detectar características locais que podem ser importantes para a detecção de faces. Finalizando o primeiro estágio, a rede retorna um único valor de saída que indica se a janela tem ou não uma face.

No segundo estágio, devido a algumas detecções falsas após o primero estágio, são utilizadas estrátegias para melhorar a confiabilidade das detecções. No artigo são apresentadas duas estrátegias: a *Merging Overlapping Detections* e a *Arbitration Among Multiple Networks*. Desta forma, o artigo mostra que foi possível obter precisão acima de 90% nas detecções, chegando até 99% em alguns casos.

2.2 Pares de Funções de Ativação e Erro

Com um foco no estudo da função de ativação *softmax* (Equação 4.6) e da função de erro *cross-entropy* (entropia cruzada) (Equações 4.12 e 4.13), Dunne and Campbell [1997] define pares naturais de funções de erro e ativação baseado em Bishop [2006], aborda uma justificativa para uso da função *softmax* e faz uma comparação experimental entre as funções de erro soma dos quadrados do erro (SSE) (Equação 4.11) e entropia cruzada.

Além das funções de erro anteriormente citadas, são usadas as funções de ativação sigmóide logística e *softmax*, fomando os seguintes pares:

- soma dos quadrados do erro e sigmóide logística (Equação 4.5);
- soma dos quadrados do erro e *softmax*;
- entropia cruzada e *softmax*.

A comparação utiliza uma simulação com 4 classes, 100 instâncias de treinamento para cada classe, 100 instâncias de teste e redes *multilayer perceptron* [Haykin, 2001] com duas

camadas, e baseia-se em dois aspectos: a precisão das estimativas da probabilidade de $P(C|x)$[1] (C representa uma classe e x uma instância) e as taxas de erro.

O trabalho de Dunne and Campbell [1997] mostra que o par *softmax* e entropia cruzada teve melhor capacidade de generalização.

2.3 Conclusões

Apesar de existirem vários trabalhos na literatura que abordam redes neurais artificais, Dunne and Campbell [1997] foi o único estudo encontrado que realiza uma comparação entre os pares de funções de erro e funções de ativação. No entanto, a comparação realizada no artigo mencionado se restringe aos pares citados na Seção 2.2, redes *multilayer perceptron* e aprendizagem através do algoritmo *back-propagation*.

Nosso trabalho propõe uma comparação experimental utilizando as funções de erro: soma dos quadrados do erro, entropia cruzada para duas classes e entropia cruzada multiclasses. As funções de ativação utilizadas são: linear, sigmóide logística e *softmax*. Além disso, utilizamos as redes de funções de base radial e um método de segunda ordem para o treinamento da rede.

[1] A Probabilidade Condicional representada por $P(A|B)$ refere-se à probabilidade de um evento A ocorrer dado que o evento B aconteceu.

3 Reconhecimento de Imagens

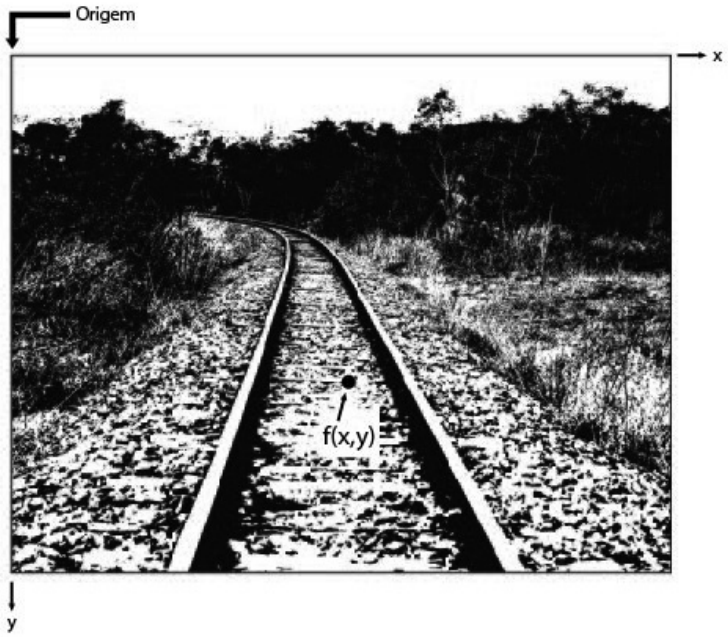
Um dos principais objetivos da análise de imagens é ser capaz de se aproximar da capacidade humana de realizar tarefas como: identificar objetos através da visão, reconhecer pessoas através da fisionomia, entre outras tarefas realizadas através da capacidade visual. Por exemplo, em um sistema para reconhecimento facial, os padrões relevantes são as diferentes faces humanas, enquanto o objetivo é obter uma precisão de reconhecimento que seja mais próxima possível da capacidade humana de realizar a respectiva tarefa. O reconhecimento de imagens é um processo de descobrir, identificar e discernir padrões relevantes em imagens [Gonzalez and Woods, 2000].

Neste capítulo, mostraremos como são representadas as imagens digitais, os passos fundamentais para o processamento de imagens e uma descrição de instâncias e classes.

3.1 Representação de Imagens Digitais

Uma imagem monocromática é uma imagem bidimensional, onde x e y representam as coordenadas espaciais. A intensidade de brilho, ou escala de cinza, em qualquer ponto é dada por uma função $f(x, y)$ [Gonzalez and Woods, 2000, Sonka et al., 2014]. Na Figura 1, podemos observar a representação de uma imagem digital monocromática.

Figura 1 – Representação de uma imagem digital.



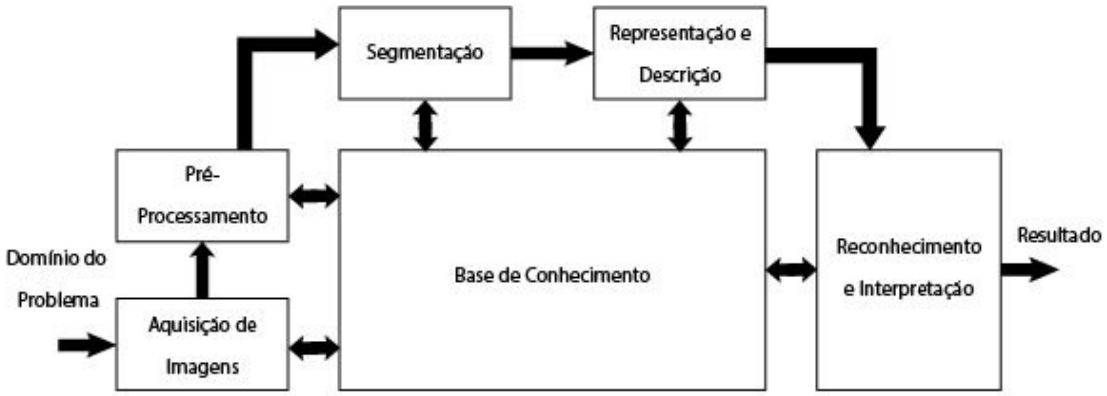
Uma imagem digital também pode ser descrita como uma matriz cujas linhas e colunas determinam um ponto na imagem e o repectivo valor determina o nível de cinza naquele ponto. Cada ponto da matriz corresponde a um elemento da imagem, também conhecido como *pixel*.

Apesar de existirem tecnologias para a obtenção de imagens digitais coloridas, esta seção tem o intuito de dar uma visão geral sobre a representação de imagens digitais e não abordará imagens coloridas.

3.2 Processamento de Imagens

O processamento de imagens abrange os seguintes passos fundamentais: Aquisição, Pré-Processamento, Segmentação, Representação e Descrição, Reconhecimento e Interpretação [Gonzalez and Woods, 2000]. Na Figura 2, observamos a sequência de passos para realizar o processamento de imagens. Na aquisição temos a obtenção das imagens através de um sensor de imageamento capaz de digitalizar uma imagem, como por exemplo: instrumentos ópticos, câmeras fotográficas, aparelhos de diagnóstico (radiografia, ressonância magnética, tomografia computadorizada, etc), scanners, entre outros.

Figura 2 – Passos fundamentais em processamento de imagens digitais.



No pré-processamento, após a obtenção das imagens, as imagens são melhoradas com técnicas como realce de contraste, remoção de ruídos e isolamento de regiões, visando aumentar a chance de sucesso dos processos seguintes. A seguir vem a segmentação, também conhecida como o primeiro passo para a análise de imagens e é um fator determinante para o sucesso ou fracasso da análise. A segmentação subdivide a imagem em partes ou objetos constituintes. Por exemplo, em aplicações para reconhecimento de veículos terrestres, o objetivo está na identificação de veículos em uma estrada. Primeiramente, é segmentada a estrada, em seguida, os elementos constituintes da estrada em objetos que possuam tamanho próximo ao de um veículo em potencial.

Na fase de representação e descrição os dados são convertidos para um forma adequada ao processamento computacional, após a escolha de uma representação (por fronteiras ou região) e uma forma de descrição (seleção de características). Por último, o estágio de reconhecimento e interpretação, onde o reconhecimento trata de atribuir um rótulo a um objeto através das informações fornecidas. E a interpretação tem a função de atribuir um significado ao conjuto de objetos reconhecidos.

Este trabalho está inserido na fase de reconhecimento, onde uma rede neural artificial será utilizada como descritora.

3.2.1 Conceitos básicos

No reconhecimento de imagens, uma instância é uma descrição quantitativa de um objeto ou entidade existente em uma imagem. Uma instância é formada por uma ou mais características, ou seja, uma instância é um arranjo de características. Uma classe de instâncias é um conjuto de padrões que têm características em comum [Gonzalez and Woods, 2000].

Vetores, cadeias e árvores são os três principais arranjos de instâncias usados [Gonzalez and Woods, 2000]. Neste trabalho utilizaremos as instâncias em forma de vetores, onde são representados por letras em negrito tomando a forma:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

onde cada componente x_i representa a i-ésima característica e n o número de características.

3.3 Conclusões

Neste capítulo, mostramos como são representadas as imagens digitais monocromáticas, os cincos passos fundamentais do processamento de imagens e os conceitos básicos de instâncias e classes aplicados ao reconhecimento de imagens. Mostramos também que este trabalho está inserido na última fase do processamento de imagens, onde se encontra a fase de reconhecimento.

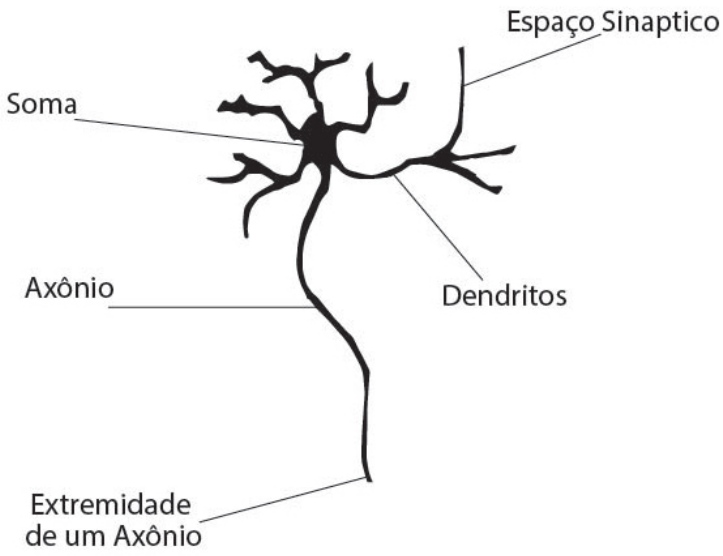
No próximo capítulo, discutiremos as redes neurais artificiais, seus componentes, arquiteturas, redes de funções de base radial e os métodos de aprendizagem.

4 Redes Neurais Artificiais

Pesquisas sobre redes neurais artificiais (RNAs) são motivadas pelo fato de que o cérebro humano processa informações de maneira totalmente diferente do computador digital, equiparando-se a um sistema de processamento de informação altamente complexo, não-linear e paralelo. Uma rede neural é formada por neurônios e sinapses, onde através de sua organização estrutural consegue um poder de processamento para certas tarefas (por exemplo: percepção, reconhecimento de padrões e controle motor) muito maior que muitos computadores digitais da atualidade [Haykin, 2001].

Um neurônio biológico é constituido de três parte principais: o corpo da célula, os dendritos e o axônio, cada um constitui funções específicas e complementares. Os dentritos recebem informações de outros neurônios e conduzem até o corpo celular, onde a informação é processada gerando novos impulsos. Assim, novos impulsos são transmitidos através do axônio até os dentritos dos neurônios aos quais estão conectados. O ponto que liga a terminação axônica de um neurônio e o dendrito de outro é chamado de sinapse. Através das sinapses, os neurônios se unem formando uma rede neural. A sinapse também é responsável por controlar a transmissão de impulsos, ou o fluxo de informação, entre as redes. Logo, a capacidade de adaptação de um neurônio vem do efeito das sinapses, que é variável [de Pádua Braga et al., 2000]. Na Figura 3, podemos visualizar uma forma simplificada de um neurônio biológico.

Figura 3 – Representação simplificada de um neurônio biológico.



Inspiradas nas redes neurais biológicas, as RNAs são sistemas paralelos divididos em unidades – neurônios artificiais – capazes de representar uma função matemática, normalmente não-linear. Essas unidades são dipostas em várias camadas e interligadas por conexões associadas a pesos em analogia às sinapses biológicas, que armazenam todo o conhecimento da rede.

Uma rede neural é um processador maciçamente e paralelamente distribuído constituído de unidades de processamento simples, que têm a capacidade natural para armazenar conhecimento experimental e torná-lo disponível para o uso [Haykin, 2001]. As RNAs se assemelham ao cérebro em dois aspectos:

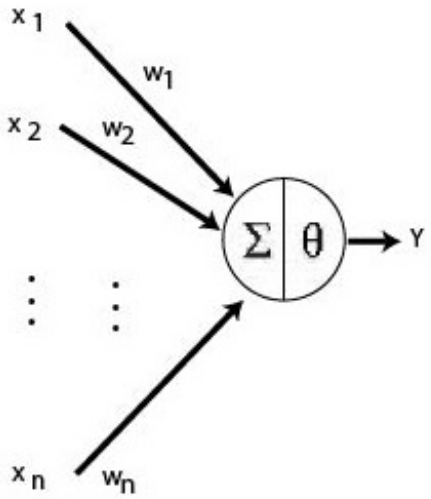
- A obtenção do conhecimento da rede acontece através da influência do seu ambiente a partir de um processo de aprendizagem;
- O seu conhecimento é armazenado através da influência variável das conexões sinápticas.

Neste capítulo, detalharemos arquiteturas e métodos de aprendizegem de RNAs. Apresentaremos a rede utilizada neste trabalho e o processo utilizado para sua aprendizagem.

4.1 Neurônio Artificial

Um neurônio é a unidade de processamento de informação de uma rede neural. Na Figura 4 podemos observar o modelo simplificado de um neurônio biológico, tal modelo foi proposto por McCulloch and Pitts [1943] (Modelo MCP). A seguir descrevemos alguns elementos básicos de um neurônio, também observados na Figura 4.

Figura 4 – Neurônio de McCulloch e Pitts.



- Um conjunto de n terminais de entrada $x_1, x_2, ..., x_n$, representando os dentritos, e um terminal de saída y, representando o axônio.
- Os terminais de entrada têm ligações com pesos $w_1, w_2, ..., w_n$, cujos valores variam durante a aprendizagem, podendo ser negativos ou positivos. Esses pesos representam o comportamento das sinapses, sendo o efeito da sinapse i no neurônio pós-sináptico dado por $x_i w_i$.
- Um somador para todas as entradas após sofrerem o efeito das respectivas sinapses do neurônio.
- Uma função de ativação θ para restringir o valor da saída. No modelo MCP essa função é dada por um limiar que, de acordo com a soma da entrada ponderada pelos respectivos pesos, decide se o neurônio deve ou não "disparar"(saída igual a 1 ou a 0).

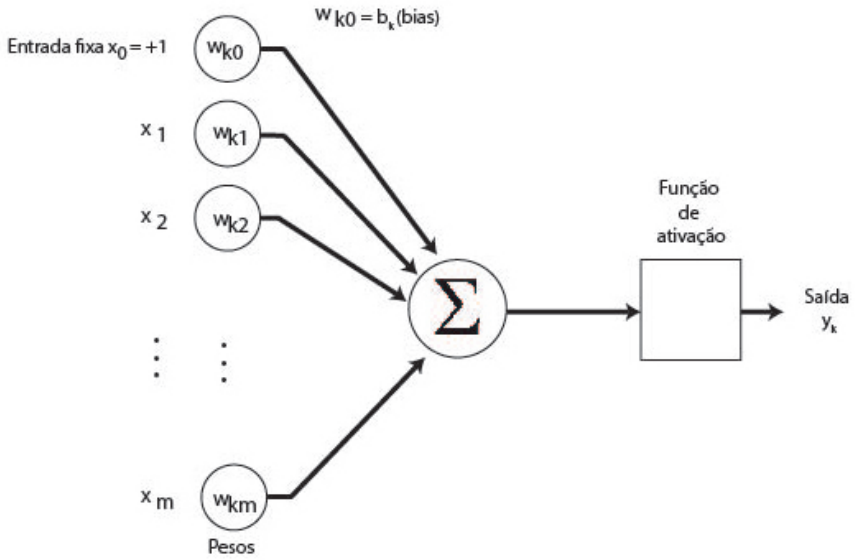
Um neurônio biológico "dispara"quando a soma dos impulsos que ele recebe excede o seu limiar de excitação [de Pádua Braga et al., 2000, Hayman, 1999]. No neurônio artificial, os impulsos são representados pelos terminais de entrada ponderados pelos respectivos pesos, o limiar de excitação é representado pela função de ativação e o disparo pelo valor de saída. Então, sabendo-se que a função de ativação do neurônio MCP é um limiar, o neurônio de McCulloch e Pitts terá sua saída ativada quando:

$$\sum_{i=1}^{n} x_i w_i \geq \theta \tag{4.1}$$

onde n é o número de entradas do neurônio, w_i é o peso associado à entrada x_i e θ é o limiar do neurônio.

Em modelos mais robustos, como o *Perceptron* [Bishop et al., 1995, Haykin, 2001] , o neurônio conta também com uma unidade especial chamada de viés (*bias*) que funciona como um peso extra, usado para aumentar o grau de liberdade, permitindo uma melhor generalização do conhecimento ao neurônio fornecido. É possível observar um exemplo do modelo de neurônio *perceptron* na Figura 5.

Figura 5 – Modelo de Neurônio *Perceptron*.



4.2 Funções de Ativação

Uma função de ativação define a saída de um neurônio baseado no somatório dos valores de entrada ponderados pelos repectivos pesos [Haykin, 2001]. Após o modelo MCP, vários modelos de neurônios foram desenvolvidos, capazes de gerar uma saída qualquer com diferentes funções de ativção [de Pádua Braga et al., 2000]. Aqui mencionamos quatro funções de ativação básicas:

- Apresentada na Figura 6a, a função **linear** é determinada pela Equação 4.2.

$$y = \alpha x \tag{4.2}$$

  onde α é um número real, y é a saída e x o valor de entrada da função.

- A função **rampa**, como mostra a Figura 6b, é uma modificação da função linear, onde a Equação 4.2 é restringida para gerar valores em uma faixa $[-\gamma, +\gamma]$ e sua forma matemática pode ser vista na Equação 4.3.

$$y = \begin{cases} +\gamma, & \text{se } x \geq +\gamma \\ x, & \text{se } |x| < +\gamma \\ -\gamma, & \text{se } x \leq -\gamma \end{cases} \tag{4.3}$$

  onde $-\gamma$ e $+\gamma$ são os valores mínimo e máximo da saída.

- A função **limiar**, ilustrada na Figura 6c, estabelece uma saída $+\gamma$ para os valores de $x > 0$ e $-\gamma$ caso contrário. Na Equação 4.4 é representada a função limiar.

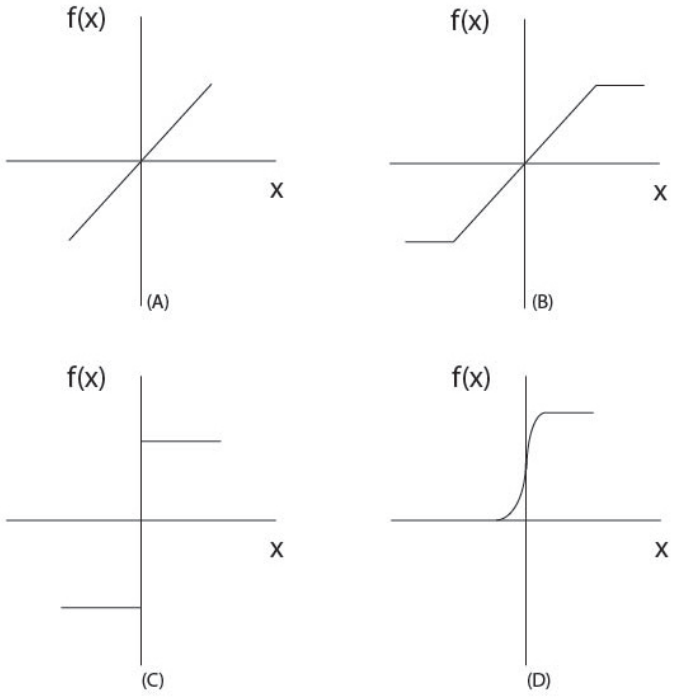
$$y = \begin{cases} +\gamma, & \text{se } x > 0 \\ -\gamma, & \text{se } x \leq 0 \end{cases} \tag{4.4}$$

- Com um gráfico em forma de S, como mostra a Figura 6d, a função **sigmoidal** é uma função semilinear, limitada e monotônica. A função logística, definida na Equação 4.5, é uma das mais relevantes funções sigmoidais [de Pádua Braga et al., 2000].

$$y = \frac{1}{1 + e^{\frac{-x}{T}}} \tag{4.5}$$

  onde T determina a suavidade da curva.

Figura 6 – (a) Função Linear. (b) Função Rampa. (c) Função Degrau. (d) Função Sigmoidal.



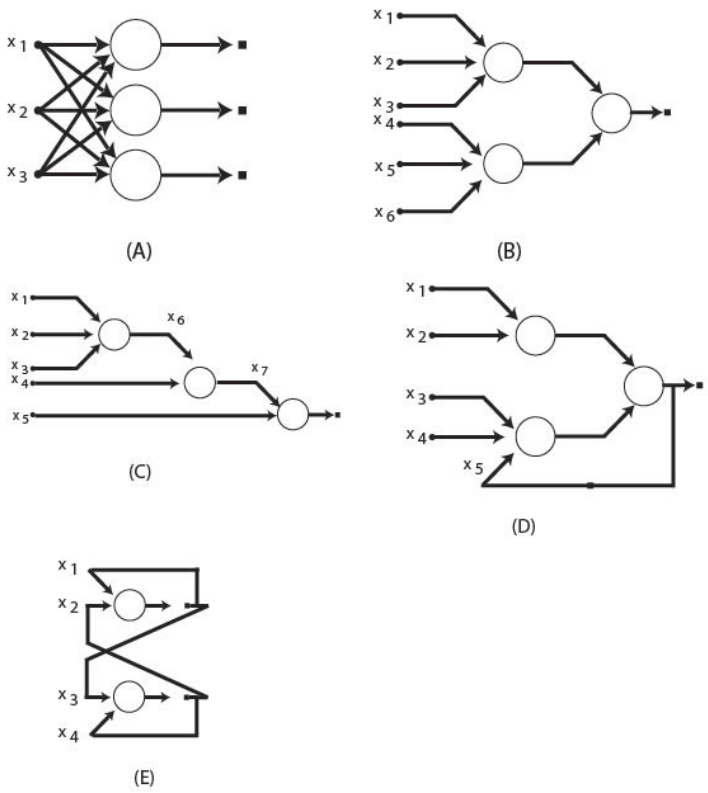
Outra função de ativação é a função *softmax* [Dunne and Campbell, 1997]. Definida na Equação 4.6, a *softmax* é uma função que produz um valor de saída em termos percentuais, em relação a saída de outros $C - 1$ neurônios, e assegura que $\sum_{k=1}^{C} y_k = 1$, onde C é o número de neurônios e y_k é o valor de saída do neurônio k.

$$y_j = \frac{e^{x_j}}{\sum_{k=1}^{C} e^{x_k}} \tag{4.6}$$

4.3 Arquitetura

A definição da arquitetura de uma rede neural conta com os seguintes parâmetros: número de camadas da rede, número de neurônios em cada camada, tipo de conexão entre os neurônios e a topologia da rede. A escolha da arquitetura de uma rede é parte relevante na sua construção. Tal seleção depende do problema a ser tratado e tem grande impacto na capacidade de generalização da RNA [de Pádua Braga et al., 2000]. Na Figura 7, são expostos alguns modelos de arquiteturas de RNAs.

Figura 7 – Modelos de arquiteturas de RNAs.



Na escolha do número de camadas da rede, tem-se as alternativas a seguir.

- Redes de Camada Única, onde encontra-se todos os neurônios organizados numa única camada entre a entrada e a saída da rede, como mostra a Figura 7(a, e);
- Redes de Múltiplas Camadas, onde encontra-se os neurônios organizados em mais de uma camada entre a entrada e a saída da rede, como visto na Figura 7(b, c, d).

Ao definir o tipo de ligação entre os neurônios, podemos optar por:

- *feedforward*, onde a rede não sofre uma realimentação. As conexões entre os neurônios não formam um ciclo, ou seja, um determinado neurônio só se conecta com o neurônio da camada anterior ou da camada seguinte. A Figura 7(a, b, c) mostra alguns exemplos;
- *feedback*, ocorre quando alguma conexão entre os neurônios da rede forma um ciclo. Na Figura 7(d, e), podemos observar alguns modelos.

Por fim, as redes também podem ser especificadas por sua conectividade.

- Redes Parcialmente Conectadas, como mostra a Figura 7(b, c, d);
- Redes Completamente Conectadas, como mostra a Figura 7(a, e).

Por serem consideradas aproximadores universais [Hornik et al., 1989], nesse trabalho utilizaremos a arquitetura de múltiplas camadas, com ligações do tipo *feedforward* e com a rede completamente conectada.

4.4 Redes de Funções de Base Radial

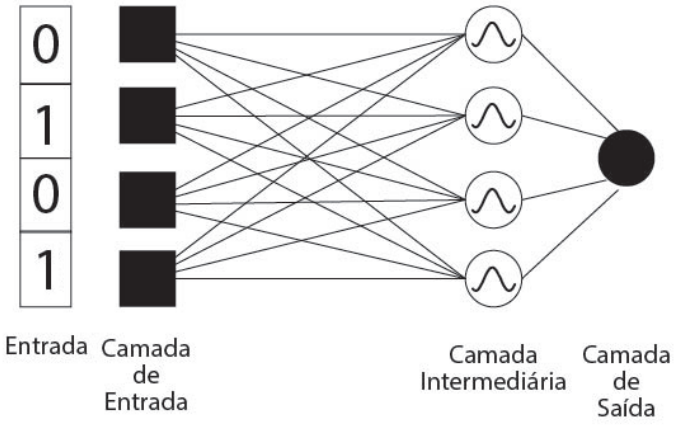
Redes de Funções de Base Radial (RBF, do inglês *Radial Basis Functions*) são redes neurais multicamadas que empregam como função de ativação, nos neurônios da camada intermediária, funções de base radial em relação à distância entre seus vetores de entrada e de peso [Bishop et al., 1995, Broomhead and Lowe, 1988, de Pádua Braga et al., 2000].

Nesta seção, mostraremos a arquitetura das redes RBF e seu treinamento.

4.4.1 Arquitetura

Redes RBF têm tipicamente uma camada escondida (intermediária) e uma camada de saída com qualquer número de nodos, onde a função da camada intermediária é agrupar os dados de entrada em conjuntos (*clusters*) e transformar a entrada não linearmente separável em um conjunto linearmente separável. Já camada de saída tem a função de classificar os dados recebidos pela camada intermediária [de Pádua Braga et al., 2000]. Na Figura 8 é ilustrado um exemplo de arquitetura de uma rede RBF.

Figura 8 – Rede RBF típica com uma camada intermediária.



Funções radiais são funções $\phi(v)$ cujos valores dependem apenas da distância entre a origem x e um ponto central c dada por $\|x - c\|$, onde a norma usualmente utilizada é a distância euclidiana [Buhmann, 2003]. As funções de base radiais mais utilizadas em redes RBF são:

- Função Gaussiana:

$$\phi(v) = e^{-\frac{v^2}{2\sigma^2}} \tag{4.7}$$

- Função Multiquadrática:

$$\phi(v) = \sqrt{(v^2 + \sigma^2)} \tag{4.8}$$

- Função *thin-plate-spline*:

$$\phi(v) = v^2 \log(v) \tag{4.9}$$

onde $v = \|x - c\|$ e σ a largura da função radial.

A reposta produzida por um nodo de saída j é definida pela Equação 4.10.

$$y_j = f_j\left(\sum_{i=1}^{K} w_{ji}\phi(\|x - c_i\|) + w_{j0}\right), \tag{4.10}$$

onde w_{ji} representa o peso da ligação entre o neurônio da camada intermediária i e o neurônio da camada de saída j, f_j a função de ativação do nodo de saída j e w_{j0} o viés.

Um fator importante numa RNA é o número de neurônios da camada intermediária. Uma forma de definir a quantidade de neurônios da camada intermediária de uma rede RBF é estabelecer o número de neurônios igual ao número de instâncias do conjunto de dados. Neste caso, os centros são determinados pelos vetores do conjunto de dados. Assim a rede mapeia com rigor os dados de entrada para a saída exata, entretanto isso pode levar ao *overfitting* [de Pádua Braga et al., 2000, Hawkins, 2004].

Uma maneira de tratar o problema de *overfitting* é definir o número de nodos menor que o número total de padrões de entrada sem restrigir os centros aos vetores de entrada. Através de métodos como o *K-means* [MacQueen et al., 1967], pode-se encontrar centros, em regiões do espaço onde estão situados os vetores mais representativos para as funções radiais [de Pádua Braga et al., 2000].

Em contrapartida as redes RBF, existem as redes *Multilayer Perceptron* (MLP)[1] [Bishop et al., 1995, de Pádua Braga et al., 2000, Haykin, 2001]. As redes MLP têm custo computacional maior que as redes RBF [de Pádua Braga et al., 2000, Haykin, 2001], motivando assim a utilização das redes RBF neste trabalho. Utilizaremos a função gaussiana nos neurônios da camada intermediária, dada pela Equação 4.7.

4.4.2 Treinamento

Embora existam outros métodos [Neruda and Kudová, 2005], as redes RBF são tipicamente treinadas em dois estágios [de Pádua Braga et al., 2000]. No primeiro estágio, o número de nodos da camada intermediária é determinado, ou seja, é definido o número de funções de base radial que serão utilizadas na rede e seus parâmetros através de métodos não supervisionados. Contudo, as instâncias de treinamento podem ser usadas como centros das funções de base radial, ocasionando o mapemento rigoroso dos dados de entrada para a saída exata, como citado na Seção 4.4.1.

No segundo estágio, os pesos dos neurônios de saída são ajustados. Os neurônios da camada de saída são tipicamente do tipo *perceptron* [Rosenblatt, 1957]. Desta forma, os pesos podem ser determinados por algoritmos como o *back-propagation* [Masson and

[1] MLP são redes multicamadas formadas por neurônios do tipo *perceptron*.

Wang, 1990, Rumelhart et al., 1988] e suas variações [Fahlman, 1988, Parma et al., 1999, Riedmiller and Braun, 1992], entre outros algoritmos utilizados para o treinamento de redes neurais.

Neste trabalho, o número de funções de base radial das redes neurais é determinado pelo número de instâncias do conjunto de treinamento[2], seus centros são determinados pelos vetores do conjunto de treinamento[3] e suas larguras são definidas por validação cruzada. O algoritmo utilizado para o treinamento da camada de saída da rede neural é baseado no Método dos Mínimos Quadrados Iterativamente Redistribuídos que será visto na Seção 4.5.4.

4.5 Aprendizagem de Redes Neurais

Uma característica essencial das redes neurais artificiais é a capacidade de aprender a partir de instâncias de treinamento, através do ajuste da intensidade das conexões entre os neurônios. Um conceito de aprendizagem pode ser expressado da seguinte forma [Mendel and McLaren, 1970]:

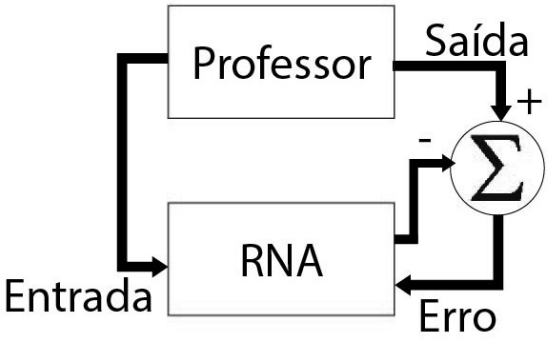
> Aprendizagem é o processo pelo qual os parâmentros de uma rede neural são ajustados através de uma forma continua de estímulo pelo ambiente no qual a rede está operando, sendo o tipo específico de aprendizagem realizada definido pela maneira particular como ocorrem os ajustes realizados nos parâmetros.

Os métodos para treinamento de uma rede neural podem ser agrupados em três paradigmas principais: aprendizado supervisionado, aprendizado não-supervisionado [Hertz et al., 2008] e aprendizado semi-supervisionado [Chapelle et al., 2006].

[2] Neste trabalho, o limite de neurônios da camada intermediária é de 100 neurônios.

[3] Neste trabalho, caso o número de instâncias de treinamento seja acima de 100 instâncias, os centros das funções de base radial são definidos através de uma escolha aleatória e estratificada de vetores do conjunto de treinamento.

Figura 9 – Aprendizado supervisionado.



No aprendizado supervisionado a resposta da rede neural é avaliada por um supervisor de acordo com a saída desejada em relação à instância de entrada atual. Os ajustes dos pesos são calculados de maneira que a saída da rede tenda a se igualar à resposta do supervisor. A Figura 9 representa o aprendizado supervisionado.

Por sua vez, no aprendizado não-supervisionado não existe um supervisor. A rede é reponsável por descobrir por si própria relações, padrões ou categorias nos dados apresentados. Um exemplo de aprendizado não-supervisionado no nosso sistema biológico são os estágios iniciais dos sistemas de visão e audição [de Pádua Braga et al., 2000]. E no aprendizado semi-supervisionado, o supervisor está presente apenas para algumas instâncias.

Neste trabalho, utilizaremos um método supervisionado para treinamento das redes neurais.

4.5.1 Modelos de Aprendizagem

Em rede neurais, existem três principais modelos de aprendizagem [Bishop, 2006]: *online*, em lote (*batch*) e o estocástico. Nos modelos *online* e estocástico, o ajuste dos pesos acontece para cada instância de treinamento apresentado a rede. O modelo online é utilizado em ambientes dinâmicos, onde o fluxo de novas instâncias é continuo. Já o modelo estocástico usa um conjunto de instâncias de treinamento estático. No entanto, pecorre o conjunto de treinamento em ordem aleatória, a fim de obter uma melhor generalização. No modelo em lote, o ajuste dos pesos ocorre após o conjunto de treinamento ser apresentado a rede. A aprendizagem em lote produz uma convergência mais estável a um erro mínimo.

Neste trabalho, utilizaremos a aprendizagem em lote.

4.5.2 Funções de Erro

Funções de erro são responsáveis por calcular o erro da rede neural para cada instância de entrada em relação as repostas da rede e seus repectivos valores de saída desejados. A função de erro comumente utilizada é a Soma dos Quadrados do Erro (SSE) [Bishop et al., 1995], dada pela Equação 4.11.

$$E = \frac{1}{2} \sum_{c=1}^{C} (y_c - d_c)^2 , \tag{4.11}$$

onde C é o número de neurônios na camada de saída, y_c e d_c a saída obtida e a saída desejada, respectivamente, do neurônio c da camada de saída.

Usamos também, como funções de erro, a Entropia Cruzada em sua forma para problemas com duas classes e a Entropia Cruzada em sua forma para problemas multiclasses [Bishop et al., 1995], dadas respectivamente pelas Equações 4.12 e 4.13.

$$E = -\sum_{c=1}^{C} d_c \cdot \ln(y_c) + (1 - d_c) \cdot \ln(1 - y_c) \tag{4.12}$$

$$E = -\sum_{c=1}^{C} d_c \cdot \ln(y_c) \tag{4.13}$$

4.5.3 *Back-propagation*

O algoritmo *back-propagation* [McClelland et al., 1986, Rumelhart et al., 1988, Werbos, 1990] é o mais conhecido para treinamento de redes MLP, podendo também ser usado para treinar a camada de saída das redes RBF.

O *back-propagation* é um algoritmo supervisionado que por meio de uma correção de erro, ajusta os pesos da rede. O algoritmo é dividido em duas fases, onde na primeira fase (fase *forward*) é gerada a saída da rede, em seguida, na segunda fase (fase *backward*) é feito o cálculo do erro através da saída da rede neural, a saída desejada para a determinada instância de entrada n e a função de erro utilizada. Logo após, ainda na fase *backward*, são ajustados os pesos.

O algorimto *back-Propagation* compreende os seguintes passos [de Pádua Braga et al., 2000]:

Algoritmo 1: Algoritmo *back-Propagation*

Entrada: Vetor de instâncias de treinamento e os parâmentros da rede.
Saída: Rede treinada.
início
  Os pesos são inicializados aleatoriamente;
  repita
    para *cada padrão de entrada X* **faça**
      //Fase *Forward*
      A entrada é apresentada à primeira camada da rede;
      para *cada camada* C_i **faça**
        Após os nodos da camada C_i calcularem suas saídas, o vetor de saída produzido servirá de entrada para a definição das saídas dos nodos da camada C_{i+1};
      fim
      A partir do vetor de saída da última camada e o respectivo vetor de saída desejada é calculado o erro da camada de saída;
      //Fase *Backward*
      repita
        Os nodos da camada atual ajustam seus pesos de forma a reduzir seus erros;
        O erro de um nodo das camadas intermediárias é calculado utilizando os erros dos nodos da camada seguinte conectados ao neurônio, ponderados pelos pesos das conexões entre eles;
      até *chegar na primeira camada intermediária, a partir da última camada*;
    fim
  até *um dos critérios de parada ser alcançado*;
fim

O *back-propagation* tem sua fundamentação baseada na regra delta [WIDROW et al., 1960]. Os ajustes dos pesos são realizados através do método do gradiente descendente que é dado por:

$$-\nabla E.$$

Podemos definir ∇E como:

$$\begin{aligned} \nabla E &= \frac{\partial E}{\partial \mathbf{w}_j} \\ &= \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial \mathbf{w}_j}, \end{aligned} \tag{4.14}$$

onde $\mathbf{w}_j$ e y_j é o vetor de peso e a saída obtida, respectivamente, do neurônio j, $z_j = \mathbf{x} \cdot \mathbf{w}_j$ e $\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial z_j}$ mede o erro do neurônio j [de Pádua Braga et al., 2000].

4.5.4 Método dos Mínimos Quadrados Iterativamente Redistribuídos

Por anos, os estudos das RNAs tenderam a se concentrar no algoritmo *back-propagation*, pois proporciona regras de ajustes locais eficientes para determinar de forma interativa os pesos de uma rede neural. No entando, existem problemas na utilização do método do gradiente descendente, como a permanência em mínimos locais e planaltos de erro [Fukumizu and Amari, 1999, Gori and Tesi, 1992]. Sendo assim, um método de primeira ordem, tais como *back-propagation* pode exigir um grande número de iterações para convergir.

Uma abordagem alternativa é a utilização de métodos de segunda ordem, a fim de encontrar o ponto mínimo da função de erro. Nesse sentindo, o método de Newton-Raphson [Bishop et al., 1995, Haykin, 2001] é um algoritmo de segunda ordem, onde o ajuste dos pesos é dado por:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \mathbf{H}^{-1}\mathbf{G} \tag{4.15}$$

onde $\mathbf{w}$ é o vetor de pesos e vieses, $\mathbf{H}$ é a matriz hessiana, t é a iteração e $\mathbf{G} = \nabla E$.

A matriz hessiana [Warner and Misra, 1998] é uma matriz bloco $\mathbf{H} = [H_{jk}]_{W \times W}$ ($j, k \in \{1, ..., W\}$ e W é o número total de pesos e vieses), onde cada bloco é dado por:

$$H_{jk} = \left[\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k} \right] \tag{4.16}$$

A direção de busca dada por $\mathbf{H}^{-1}\mathbf{G}$, leva o processo de otimização para um ponto estacionário[4], o que não acontece no método do gradiente descendente. Portanto, o uso da regra de Newton-Raphson, muitas vezes, leva a um menor número de iterações na busca de um ponto estacionário. No entanto, embora a direção de busca aponte sempre para um ponto estacionário, este ponto é um mínimo apenas quando a matriz Hessiana é uma matriz positiva definida. Caso contrário, a regra de Newton-Raphson apontará para um ponto de sela[5] ou um máximo. Um outro problema que limita o uso do método de Newton-Raphson é a dimensionalidade $W \times W$ da matriz hessiana, em que o cálculo de $\mathbf{H}$ é uma operação $O(NW^2)$ onde N é o número de instâncias de treinamento e o cálculo da inversa é uma operação $O(W^3)$ [Bishop et al., 1995, Warner and Misra, 1998].

Baseado no método de Newton-Raphson, o algoritmo IRLS utilizado neste trabalho corresponde aos seguintes passos:

[4] Um ponto estacionário é um ponto no domínio de uma função onde a primeira derivada é nula.

[5] Um ponto de sela é um ponto estacionário, mas não se trata de um ponto de mínimo ou um ponto de máximo.

Algoritmo 2: Algoritmo IRLS

Entrada: Vetor de instâncias de treinamento e os parâmentros da rede.
Saída: Rede treinada.
início

Os pesos são inicializados aleatoriamente;

repita

Os dados são apresentados a rede e obtém-se a saída para cada padrão de entrada;

O erro é computado para cada padrão de entrada a partir dos dados de saída e seus repectivos valores desejados;

A regra de Newton-Raphson $\Delta\mathbf{w} = \mathbf{H}^{-1}\mathbf{G}$ é calculada;

η assume o valor 1;

repita

Os novos pesos são calculados como segue:

$$\mathbf{w}^{t+1} = \mathbf{w}^{t} - \eta\Delta\mathbf{w},$$

onde η é a taxa de aprendizagem;

O novo erro é calculado;

Um novo η é calculado tal que $\eta = \frac{\eta}{2}$;

até *o erro decrescer ou* η *assumir um valor mínimo*;

até *um dos critérios de parada ser alcançado*;

fim

4.5.5 Regularização

O número de neurônios na camada escondida numa rede neural é um parâmetro de livre definição, onde seu ajuste é feito com a intenção de se ter um melhor desempenho preditivo. Um valor ideal de nodos na camada escondida fornece a melhor performance de generalização.

Na prática, uma abordagem utilizada para a escolha do número de nodos M na camada oculta é feita através da variação de M e a avaliação do desempenho da rede para cada valor que M assumir. Comumente é escolhido o valor de M que leva a rede neural à melhor generalização na validação.

No entanto, há outras formas de controlar a complexidade da rede e evitar o *overfitting*.

Uma abordagem alternativa é assumir um valor relativamente grande para M e adicionar um termo de regularização para a função de erro [Bishop, 2006]. Um simples regularizador é o método *weight decay*, onde a regularização é dada por:

$$E^{t+1}(n) = E^{t}(n) + \alpha \frac{\mathbf{w}^T \mathbf{w}}{2} \tag{4.17}$$

onde $\mathbf{w}$ é o vetor de pesos e vieses e o coeficiente α regula a importância do termo de regularização em relação à função de erro.

Como foi descrito na Seção 4.4, estabelecemos o número de neurônios da camada escondida como sendo igual a quantidade de instâncias de treinamento, podendo gerar *overfitting*. No entanto, foi adicionado o termo de regularização visto nesta seção para suavizar a fronteira de decisão e resolver o problema anteriormente mencionado.

4.6 Derivações

Para realizar a aprendizagem baseada no IRLS, vista na Seção 4.5.4, é indispensável a obtenção do gradiente e da matriz hessiana da função de erro em relação ao vetor de pesos e vieses. Nesta seção, apresentaremos todas as derivadas necessárias, estudando os pares de funções de erro e ativação utilizados neste trabalho. Nós elaboramos todas as derivadas apresentadas nesta seção.

4.6.1 Entropia Cruzada e *Softmax*

Tendo a função de erro entropia cruzada descrita na Equação 4.18.

$$E = -\sum_{c=1}^{C} ln(y_c) \cdot d_c, \tag{4.18}$$

onde C é o número de classes ou nós de saída, y_c representa a saída obtida no nó c e d_c representa a saída desejada no respectivo nó.

Definida na Equação 4.19 está a função de ativação *softmax*.

$$y_j = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}}, \tag{4.19}$$

onde $z_j = \mathbf{x} \cdot \mathbf{w}_j$, sendo $\mathbf{x}$ o vetor de entrada e $\mathbf{w}_j$ o vetor de pesos e vieses da conexão entre neurônio j e o vetor de entrada.

A primeira derivada parcial do erro para determinar o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$, pode ser obtida com a regra da cadeia da seguinte forma [Dunne and Campbell, 1997]:

$$\frac{\partial E}{\partial \mathbf{w}_j} = \left(\sum_{i=1}^{C} \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_j} \right) \cdot \frac{\partial z_j}{\partial \mathbf{w}_j} + \frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2} \right)}{\partial \mathbf{w}_j}$$

A solução do primeiro fator do lado direito é descrita como:

$$\begin{aligned} \frac{\partial E}{\partial y_i} &= \frac{\partial \left(-\sum_{c=1}^{C} ln(y_c) \cdot d_c \right)}{\partial y_i} \\ &= -\frac{d_i}{y_i}. \end{aligned}$$

Em seguida, o segundo fator pode ser definido como:

$$\frac{\partial y_i}{\partial z_j} = \frac{\partial \left(\frac{e^{z_i}}{\sum_{k=1}^{C} e^{z_k}} \right)}{\partial z_j},$$

onde, para $i = j$ tem-se

$$\frac{\partial y_i}{\partial z_j} = \frac{e^{z_i} \cdot \sum_{k=1}^{C} e^{z_k} - e^{z_i} \cdot e^{z_i}}{\left(\sum_{k=1}^{C} e^{z_k} \right)^2} = y_i - y_i^2.$$

Como $i = j$, podemos reescreve $\frac{\partial y_i}{\partial z_j}$ da seguinte forma:

$$\frac{\partial y_i}{\partial z_j} = y_i(1 - y_j).$$

Em seguida, para $i \neq j$ tem-se:

$$\frac{\partial y_i}{\partial z_j} = \frac{0 \cdot \sum_{k=1}^{C} e^{z_k} - e^{z_i} \cdot e^{z_j}}{\left(\sum_{k=1}^{C} e^{z_k} \right)^2} = -y_i y_j$$

Logo, tem-se a seguinte solução para $\frac{\partial y_i}{\partial z_j}$:

$$\frac{\partial y_i}{\partial z_j} = \begin{cases} y_i(1 - y_j), & \text{se } i = j \\ -y_i y_j, & \text{se } i \neq j, \end{cases}$$

que também pode ser reescrito como

$$\frac{\partial y_i}{\partial z_j} = y_i(\delta_{ij} - y_j),$$

onde $\delta_{ij} = \begin{cases} 1, & \text{se } i = j \\ 0, & \text{se } i \neq j \end{cases}$ é o delta de Kronecker.

E, finalmente, o terceiro fator é

$$\frac{\partial z_j}{\partial \mathbf{w}_j} = \frac{\partial(\mathbf{x} \cdot \mathbf{w}_j)}{\partial \mathbf{w}_j} = \mathbf{x}.$$

A derivada parcial do termo de regularização é

$$\frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2}\right)}{\partial \mathbf{w}_j} = \alpha \mathbf{w}_j$$

Então, a solução para o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$ é descrita como segue:

$$\begin{aligned} \tfrac{\partial E}{\partial \mathbf{w}_j} &= -\textstyle\sum_{i=1}^{C} \left[\frac{d_i}{y_i} \cdot y_i \cdot (\delta_{ij} - y_j)\right] \cdot \mathbf{x} \\ &= -\left[\textstyle\sum_{i=1}^{C} d_i \delta_{ij} - \sum_{i=1}^{C} d_i y_j\right], \end{aligned}$$

sendo $\sum_{i=1}^{C} d_i = 1$, tem-se

$$\frac{\partial E}{\partial \mathbf{w}_j} = (y_j - d_j) \cdot \mathbf{x} + \alpha \mathbf{w}_j. \tag{4.20}$$

Para o método de Newton-Raphson, também deve ser determinada a segunda derivada parcial $\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}$, onde tem-se:

$$\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k} = \left(\sum_{l=1}^{C} \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_l} \cdot \frac{\partial y_l}{\partial z_k}\right) \cdot \frac{\partial z_k}{\partial \mathbf{w}_k} + \frac{\partial (\alpha \mathbf{w}_j)}{\partial \mathbf{w}_k}$$

sabendo que

$$\sum_{l=1}^{C} \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_l} \cdot \frac{\partial y_l}{\partial z_k} = \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_j} \cdot \frac{\partial y_j}{\partial z_k},$$

podemos reescrever

$$\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}$$

da seguinte maneira:

$$\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k} = \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_j} \cdot \frac{\partial y_j}{\partial z_k} \cdot \frac{\partial z_k}{\partial \mathbf{w}_k} + \frac{\partial \left(\alpha \mathbf{w}_j\right)}{\partial \mathbf{w}_k}$$

O primeiro fator do lado direito é definido como:

$$\begin{aligned} \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_j} &= \frac{\partial((y_j - d_j)\cdot \mathbf{x} + \alpha \mathbf{w}_j)}{\partial y_j} \\ &= \mathbf{x}. \end{aligned}$$

O segundo fator torna-se

$$\frac{\partial y_j}{\partial z_k} = y_j(\delta_{jk} - y_k).$$

O terceiro fator é

$$\frac{\partial z_k}{\partial \mathbf{w}_k} = \mathbf{x}.$$

E, finalmente, o termo de regularização para $j = k$ é

$$\frac{\partial \alpha \mathbf{w}_j}{\partial \mathbf{w}_k} = \alpha,$$

para $j \neq k$ temos

$$\frac{\partial \alpha \mathbf{w}_j}{\partial \mathbf{w}_k} = 0.$$

Logo, $\frac{\partial \alpha \mathbf{w}_j}{\partial \mathbf{w}_k}$ pode ser reescrito como

$$\frac{\partial \alpha \mathbf{w}_j}{\partial \mathbf{w}_k} = \alpha \cdot \delta_{jk}.$$

onde $\delta_{jk} = \begin{cases} 1, & \text{se } j = k \\ 0, & \text{se } j \neq k \end{cases}$ é o delta de Kronecker.

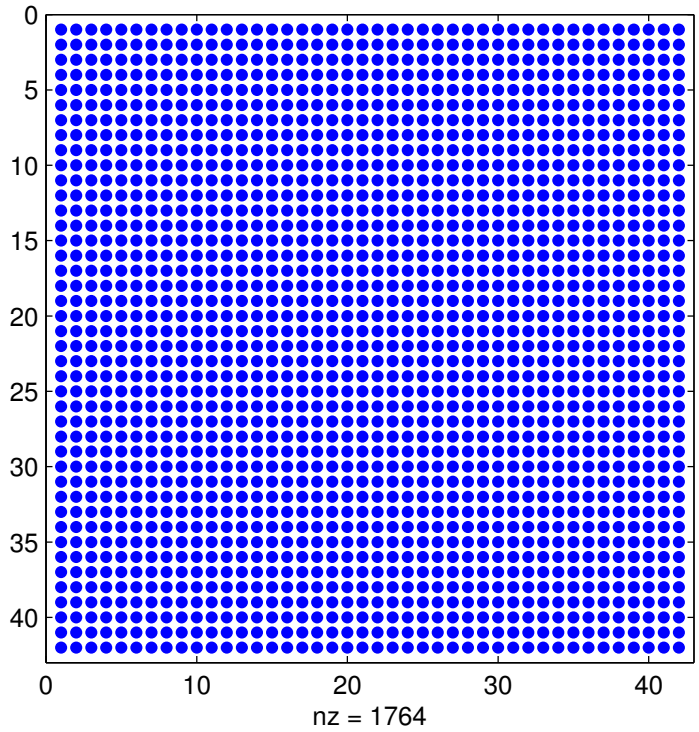
Então, como visto na Seção 4.5.4, a matriz hessiana é uma matriz bloco, onde cada bloco é

$$H_{jk} = \left[\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}\right] = \sum_{n=1}^{N} \left\{ y_{nj} \cdot (\delta_{jk} - y_{nk}) \cdot \mathbf{x}_n^T \mathbf{x}_n + \alpha \cdot \delta_{jk} \right\}, \tag{4.21}$$

onde N é o número de instâncias do conjunto de dados.

Na Figura 10, podemos observar o preenchimento da matriz hessiana gerada pelo par de função de erro entropia cruzada e função de ativação *softmax*.

Figura 10 – Preenchimento da matriz hessiana gerada pelo par de função de erro entropia cruzada e função de ativação *softmax*, base de dados com seis classes e rede neural com seis neurônios na camada escondida.



4.6.2 Entropia Cruzada e Linear

Sendo a função de erro entropia cruzada como descrita na Equação 4.18 e a função de ativação linear definida na Equação 4.22.

$$y_j = z_j, \tag{4.22}$$

onde $z_j = \mathbf{x} \cdot \mathbf{w}_j$, sendo $\mathbf{x}$ o vetor de entrada e $\mathbf{w}_j$ o vetor de pesos e vieses da conexão entre neurônio j e o vetor de entrada.

A primeira derivada parcial do erro para obter o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$, pode ser obtida com a regra da cadeia da seguinte forma:

$$\frac{\partial E}{\partial \mathbf{w}_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial \mathbf{w}_j} + \frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2} \right)}{\partial \mathbf{w}_j}$$

A solução do primeiro fator do lado direito é descrita como:

$$\begin{aligned} \frac{\partial E}{\partial y_j} &= \frac{\partial \left(- \sum_{c=1}^{C} ln(y_c) \cdot d_c \right)}{\partial y_j} \\ \\ &= -\frac{d_j}{y_j}. \end{aligned}$$

Em seguida, o segundo fator pode ser determinado como:

$$\frac{\partial y_j}{\partial z_j} = 1.$$

E, finalmente, o terceiro fator é

$$\frac{\partial z_j}{\partial \mathbf{w}_j} = \frac{\partial (\mathbf{x} \cdot \mathbf{w}_j)}{\partial \mathbf{w}_j} = \mathbf{x}.$$

O gradiente do termo de regularização é

$$\frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2} \right)}{\partial \mathbf{w}_j} = \alpha \mathbf{w}_j$$

Então, a solução para o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$ é

$$\frac{\partial E}{\partial \mathbf{w}_j} = -\frac{d_j}{y_j} \cdot \mathbf{x} + \alpha \mathbf{w}_j. \tag{4.23}$$

Para o método de Newton-Raphson, também deve ser determinada a segunda derivada parcial $\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}$, onde tem-se

$$\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k} = \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} \cdot \frac{\partial y_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial \mathbf{w}_k} + \frac{\partial (\alpha \mathbf{w}_j)}{\partial \mathbf{w}_k}$$

O primeiro fator do lado direito, para $j = k$, é definido como:

$$\begin{aligned}\frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} &= \frac{\partial\left(-\frac{d_j}{y_j}\cdot\mathbf{x}+\alpha\mathbf{w}_j\right)}{\partial y_k} \\ &= \frac{d_j\cdot\mathbf{x}}{y_j^2}.\end{aligned}$$

Para $j \neq k$

$$\frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} = 0.$$

O segundo fator torna-se

$$\frac{\partial y_k}{\partial z_k} = 1.$$

O terceiro fator é

$$\frac{\partial z_k}{\partial \mathbf{w}_k} = \mathbf{x}.$$

E, finalmente, o termo de regularização para $j = k$ é

$$\frac{\partial \alpha\mathbf{w}_j}{\partial \mathbf{w}_k} = \alpha,$$

em seguida, para $j \neq k$

$$\frac{\partial \alpha\mathbf{w}_j}{\partial \mathbf{w}_k} = 0.$$

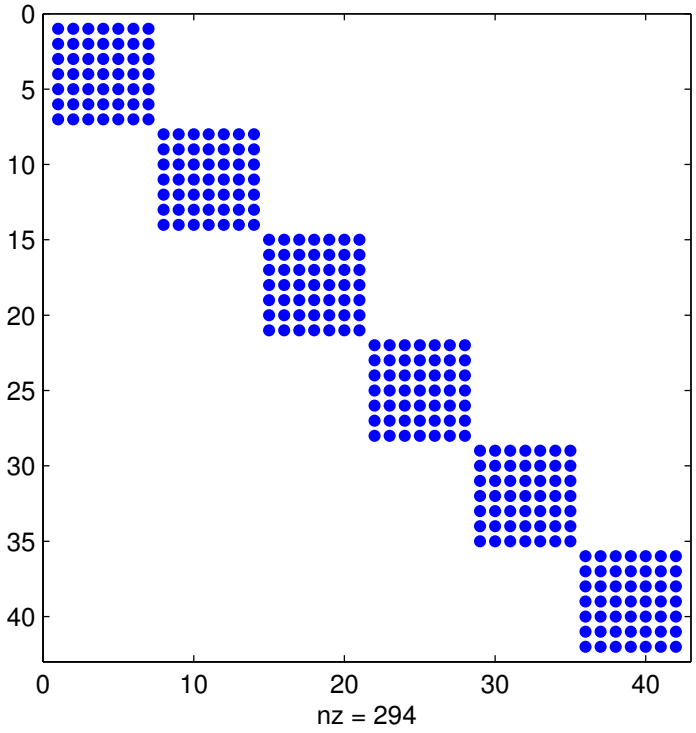
Então, como visto na Seção 4.5.4, a matriz hessiana é uma matriz bloco, onde cada bloco é

$$H_{jk} = \left[\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}\right] = \begin{cases} \sum_{n=1}^{N}\left\{\frac{d_{nk}}{y_{nk}^2}\cdot\mathbf{x}_n^T\mathbf{x}_n + \alpha\right\}, & \text{se } j = k \\ 0, & \text{se } j \neq k, \end{cases} \tag{4.24}$$

onde N é o número de instâncias do conjunto de dados.

Na Figura 11, podemos observar o preenchimento da matriz hessiana gerada pelo par de função de erro entropia cruzada e função de ativação linear.

Figura 11 – Preenchimento da matriz hessiana gerada pelo par de função de erro entropia cruzada e função de ativação linear, base de dados com seis classes e rede neural com seis neurônios na camada escondida.



4.6.3 Entropia Cruzada e Sigmóide Logística

Dispondo da função de erro entropia cruzada definida na Equação 4.18 e da função de ativação sigmóide logística definida como:

$$y_j = \frac{1}{1 + e^{-z_j}}, \tag{4.25}$$

onde $z_j = \mathbf{x} \cdot \mathbf{w}_j$, sendo $\mathbf{x}$ o vetor de entrada e $\mathbf{w}_j$ o vetor de pesos e vieses da conexão entre neurônio j e o vetor de entrada.

A primeira derivada parcial do erro para obter o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$, pode ser obtida com a regra da cadeia da seguinte forma:

$$\frac{\partial E}{\partial \mathbf{w}_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial \mathbf{w}_j} + \frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2} \right)}{\partial \mathbf{w}_j}$$

A solução do primeiro fator do lado direito, como visto na Seção 4.6.2, é descrita como:

$$\begin{aligned}\frac{\partial E}{\partial y_j} &= \frac{\partial\left(-\sum_{c=1}^{C} ln(y_c)\cdot d_c\right)}{\partial y_j} \\ &= -\frac{d_j}{y_j}.\end{aligned}$$

Em seguida, o segundo fator pode ser definido como:

$$\frac{\partial y_j}{\partial z_j} = \frac{e^{-z_j}}{\left(1+e^{-z_j}\right)^2},$$

sabendo que $y_j = \frac{1}{1+e^{-z_j}}$, obtem-se a igualdade $y_j \cdot e^{-z_j} = 1 - y_j$. Então, $\frac{\partial y_j}{\partial z_j}$ pode ser reescrito da seguinte forma:

$$\begin{aligned}\tfrac{\partial y_j}{\partial z_j} &= y_j^2 \cdot e^{-z_j} \\ &= y_j\left(1-y_j\right).\end{aligned}$$

E, finalmente, o terceiro fator é

$$\frac{\partial z_j}{\partial \mathbf{w}_j} = \frac{\partial(\mathbf{x}\cdot\mathbf{w}_j)}{\partial \mathbf{w}_j} = \mathbf{x}$$

O gradiente do termo de regularização é

$$\frac{\partial\left(\alpha\frac{\mathbf{w}^T\mathbf{w}}{2}\right)}{\partial \mathbf{w}_j} = \alpha\mathbf{w}_j$$

Então, a solução para o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$ é descrita como segue:

$$\frac{\partial E}{\partial \mathbf{w}_j} = (d_j \cdot y_j - d_j)\cdot\mathbf{x} + \alpha\mathbf{w}_j. \tag{4.26}$$

Para o método de Newton-Raphson, também deve ser determinada a segunda derivada parcial $\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}$, onde tem-se

$$\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k} = \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k}\cdot\frac{\partial y_k}{\partial z_k}\cdot\frac{\partial z_k}{\partial \mathbf{w}_k} + \frac{\partial\left(\alpha\mathbf{w}_j\right)}{\partial \mathbf{w}_k}$$

O primeiro fator do lado, para $j = k$, direito é definido como:

$$\begin{aligned} \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} &= \frac{\partial((d_j \cdot y_j - d_j) \cdot \mathbf{x} + \alpha \mathbf{w}_j)}{\partial y_k} \\ &= d_j \cdot \mathbf{x}. \end{aligned}$$

Para $j \neq k$:

$$\frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} = 0.$$

O segundo fator torna-se

$$\frac{\partial y_k}{\partial z_k} = y_k \left(1 - y_k\right).$$

O terceiro fator é

$$\frac{\partial z_k}{\partial \mathbf{w}_k} = \mathbf{x}.$$

E, finalmente, para $j = k$ o termo de regularização é

$$\frac{\partial \alpha \mathbf{w}_j}{\partial \mathbf{w}_k} = \alpha,$$

em seguida, para $j \neq k$

$$\frac{\partial \alpha \mathbf{w}_j}{\partial \mathbf{w}_k} = 0.$$

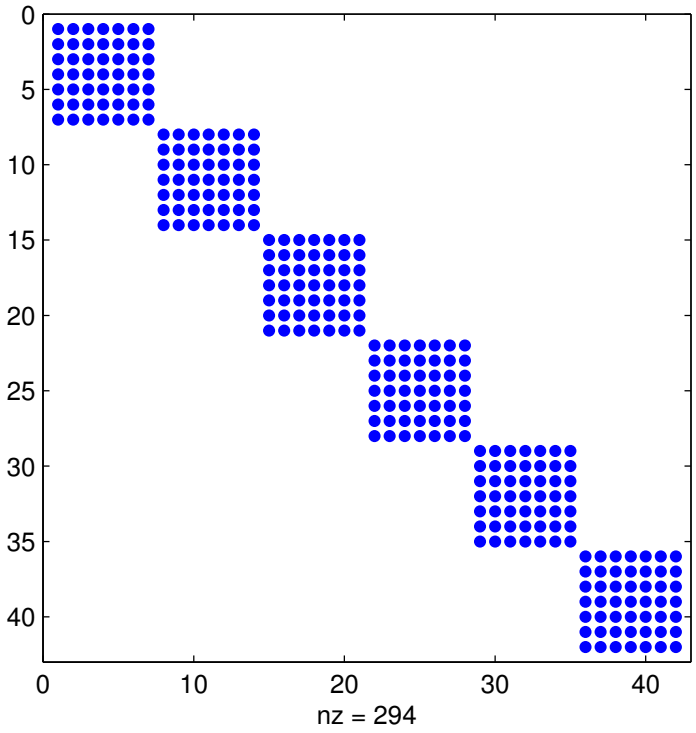
Então, como visto na Seção 4.5.4, a matriz hessiana é uma matriz bloco, onde cada bloco é

$$H_{jk} = \left[\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}\right] = \begin{cases} \sum_{n=1}^{N} \left\{(y_{nk} \cdot d_{nk} - y_{nk}^2 \cdot d_{nk}) \cdot \mathbf{x}_n^T \mathbf{x}_n + \alpha\right\}, & \text{se } j = k \\ 0, & \text{se } j \neq k, \end{cases} \tag{4.27}$$

onde N é o número de instâncias do conjunto de dados.

Na Figura 12, podemos observar o preenchimento da matriz hessiana gerada pelo par de função de erro entropia cruzada e função de ativação sigmóide logística.

Figura 12 – Preenchimento da matriz hessiana gerada pelo par de função de erro entropia cruzada e função de ativação sigmóide logística, base de dados com seis classes e rede neural com seis neurônios na camada escondida.



4.6.4 Soma dos Quadrados do Erro e *Softmax*

Sendo a função de erro SSE definida na Equação 4.28.

$$E = \frac{1}{2}\sum_{c=1}^{C}(y_c - d_c)^2, \tag{4.28}$$

onde C é o número de classes ou nós de saída, y_c representa a saída obtida no nó c e d_c representa a saída desejada no respectivo nó.

Definida na Equação 4.19 está a função de ativação *softmax*.

A primeira derivada parcial do erro para obter o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$, pode ser obtida com a regra da cadeia da seguinte forma:

$$\frac{\partial E}{\partial \mathbf{w}_j} = \left(\sum_{i=1}^{C} \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_j} \right) \cdot \frac{\partial z_j}{\partial \mathbf{w}_j} + \frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2} \right)}{\partial \mathbf{w}_j}$$

A solução do primeiro fator do lado direito é descrita como:

$$\begin{aligned} \frac{\partial E}{\partial y_i} &= \frac{\partial \left(\frac{1}{2} \sum_{c=1}^{C} (y_c - d_c)^2 \right)}{\partial y_i} \\ &= y_i - d_i. \end{aligned}$$

Em seguida, o segundo fator pode ser determinado como visto na Seção 4.6.1:

$$\frac{\partial y_i}{\partial z_j} = y_i(\delta_{ij} - y_j),$$

onde $\delta_{ij} = \begin{cases} 1, & \text{se } i = j \\ 0, & \text{se } i \neq j \end{cases}$ é o delta de Kronecker.

E, finalmente, o terceiro fator é

$$\frac{\partial z_j}{\partial \mathbf{w}_j} = \frac{\partial (\mathbf{x} \cdot \mathbf{w}_j)}{\partial \mathbf{w}_j} = \mathbf{x}$$

O gradiente do termo de regularização é

$$\frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2} \right)}{\partial \mathbf{w}_j} = \alpha \mathbf{w}_j$$

Então, a solução para o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$ é definida como segue:

$$\frac{\partial E}{\partial \mathbf{w}_j} = \sum_{i=1}^{C} \left[(y_i^2 - d_i \cdot y_i) \cdot (\delta_{ij} - y_j) \right] \cdot \mathbf{x} + \alpha \mathbf{w}_j. \tag{4.29}$$

Para o método de Newton-Raphson, também deve ser determinada a segunda derivada parcial $\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}$, onde tem-se

$$\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k} = \left(\sum_{l=1}^{C} \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_l} \cdot \frac{\partial y_l}{\partial z_k} \right) \cdot \frac{\partial z_k}{\partial \mathbf{w}_k} + \frac{\partial (\alpha \mathbf{w}_j)}{\partial \mathbf{w}_k}$$

O primeiro fator do lado direito, para $l = j$, é definido como:

$$\begin{aligned}\frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_l} &= \frac{\partial\left(\sum_{i=1}^{C}\left[(y_i^2 - d_i \cdot y_i)\cdot(\delta_{ij} - y_j)\right]\cdot\mathbf{x} + \alpha\mathbf{w}_j\right)}{\partial y_l} \\ &= [y_{nl}\,(d_{nl} - 2y_{nl}) + 2y_{nl} - d_{nl}] \cdot \mathbf{x}.\end{aligned}$$

Para $l \neq j$, tem-se

$$\frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_l} = [(d_l - 2y_l) \cdot y_j] \cdot \mathbf{x}.$$

O segundo fator torna-se

$$\frac{\partial y_l}{\partial z_k} = y_l(\delta_{lk} - y_k).$$

O terceiro fator é

$$\frac{\partial z_k}{\partial \mathbf{w}_k} = \mathbf{x}.$$

E, finalmente, para $j = k$ o termo de regularização é

$$\frac{\partial \alpha\mathbf{w}_j}{\partial \mathbf{w}_k} = \alpha,$$

em seguida, para $j \neq k$

$$\frac{\partial \alpha\mathbf{w}_j}{\partial \mathbf{w}_k} = 0.$$

Logo, $\frac{\partial \alpha\mathbf{w}_j}{\partial \mathbf{w}_k}$ pode ser reescrito como visto na Seção 4.6.1:

$$\frac{\partial \alpha\mathbf{w}_j}{\partial \mathbf{w}_k} = \alpha \cdot \delta_{jk}.$$

Então, como visto na Seção 4.5.4, a matriz hessiana é uma matriz bloco, onde cada bloco

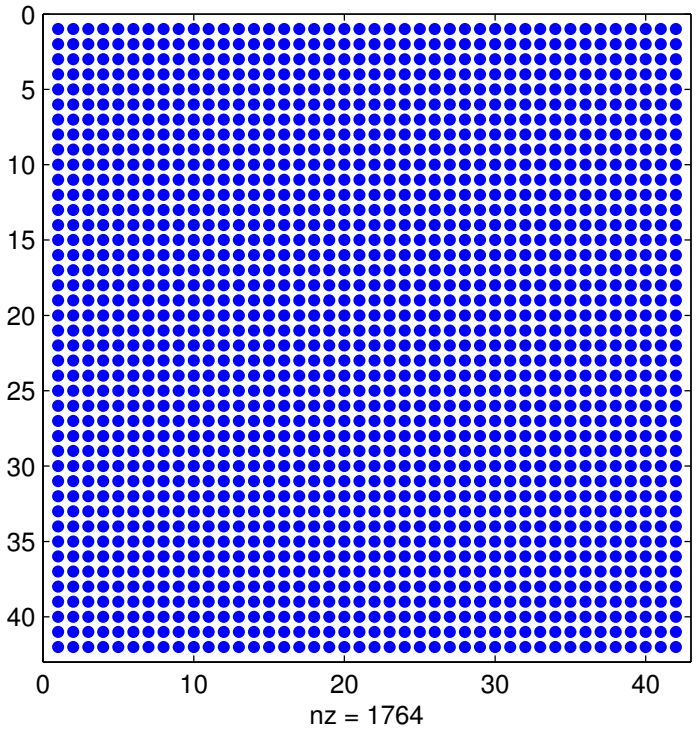
é

$$
\begin{aligned}
H_{jk} &= \left[\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}\right] \\
&= \sum_{n=1}^{N} \left\{ \left[\sum_{l=1}^{C} \begin{cases} [y_{nl}\,(d_{nl} - 2y_{nl}) + 2y_{nl} - d_{nl}] \cdot y_{nl} \cdot (\delta_{lk} - y_{nk}), & \text{se } l = j \\ (2y_{nl} - d_{nl}) \cdot -y_{nj} \cdot y_{nl} \cdot (\delta_{lk} - y_{nk}), & \text{se } l \neq j \end{cases} \right] \right. \\
&\quad \left. \cdot \mathbf{x}_n^T \mathbf{x}_n + \alpha \cdot \delta_{jk} \right\},
\end{aligned} \tag{4.30}
$$

onde N é o número de instâncias do conjunto de dados.

Na Figura 13, podemos observar o preenchimento da matriz hessiana gerada pelo par de função de erro SSE e função de ativação *softmax*.

Figura 13 – Preenchimento da matriz hessiana gerada pelo par de função de erro SSE e função de ativação *softmax*, base de dados com seis classes e rede neural com seis neurônios na camada escondida.



4.6.5 Soma dos Quadrados do Erro e Linear

Dada a função de erro SSE definida na Equação 4.28 e a função de ativação linear definida na Equação 4.22.

A primeira derivada parcial do erro para obter o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$, pode ser obtido com a regra da cadeia da seguinte forma:

$$\frac{\partial E}{\partial \mathbf{w}_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial \mathbf{w}_j} + \frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2}\right)}{\partial \mathbf{w}_j}$$

A solução do primeiro fator do lado direito é definida como:

$$\begin{aligned} \frac{\partial E}{\partial y_j} &= \frac{\partial \left(\frac{1}{2} \sum_{c=1}^{C} (y_c - d_c)^2\right)}{\partial y_j} \\ &= y_j - d_j. \end{aligned}$$

Em seguida, o segundo fator pode ser definido como:

$$\frac{\partial y_j}{\partial z_j} = 1,$$

E, finalmente, o terceiro fator é

$$\frac{\partial z_j}{\partial \mathbf{w}_j} = \frac{\partial (\mathbf{x} \cdot \mathbf{w}_j)}{\partial \mathbf{w}_j} = \mathbf{x}$$

O gradiente do termo de regularização é

$$\frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2}\right)}{\partial \mathbf{w}_j} = \alpha \mathbf{w}_j$$

Então, a solução para o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$ é descrita como segue:

$$\frac{\partial E}{\partial \mathbf{w}_j} = (y_j - d_j) \cdot \mathbf{x} + \alpha \mathbf{w}_j. \tag{4.31}$$

Para o método de Newton-Raphson, também deve ser determinada a segunda derivada parcial $\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}$, onde tem-se

$$\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k} = \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} \cdot \frac{\partial y_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial \mathbf{w}_k} + \frac{\partial \left(\alpha \mathbf{w}_j\right)}{\partial \mathbf{w}_k}$$

O primeiro fator do lado direito, para $j = k$, é descrito como:

$$\begin{aligned} \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} &= \frac{\partial((y_j - d_j) \cdot \mathbf{x} + \alpha \mathbf{w}_j)}{\partial y_k} \\ &= \mathbf{x}. \end{aligned}$$

Para $j \neq k$, tem-se

$$\frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} = 0.$$

O segundo fator torna-se

$$\frac{\partial y_k}{\partial z_k} = 1.$$

O terceiro fator é

$$\frac{\partial z_k}{\partial \mathbf{w}_k} = \mathbf{x}.$$

E, finalmente, para $j = k$ o termo de regularização é

$$\frac{\partial \alpha \mathbf{w}_j}{\partial \mathbf{w}_k} = \alpha,$$

em seguida, para $j \neq k$

$$\frac{\partial \alpha \mathbf{w}_j}{\partial \mathbf{w}_k} = 0.$$

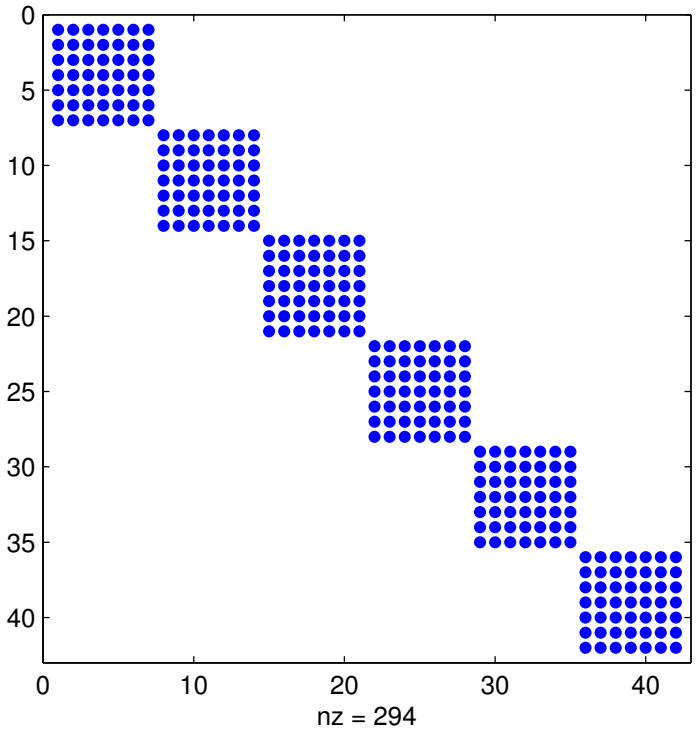
Então, como visto na Seção 4.5.4, a matriz hessiana é uma matriz bloco, onde cada bloco é

$$H_{jk} = \left[\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}\right] = \begin{cases} \sum_{n=1}^{N} \left\{\mathbf{x}_n^T \mathbf{x}_n + \alpha\right\}, & \text{se } j = k \\ \\ 0, & \text{se } j \neq k, \end{cases} \tag{4.32}$$

onde N é o número de instâncias do conjunto de dados.

Na Figura 14, podemos observar o preenchimento da matriz hessiana gerada pelo par de função de erro SSE e função de ativação linear.

Figura 14 – Preenchimento da matriz hessiana gerada pelo par de função de erro SSE e função de ativação linear, base de dados com seis classes e rede neural com seis neurônios na camada escondida.



4.6.6 Soma dos Quadrados do Erro e Sigmóide Logística

Tendo a função de erro SSE definida na Equação 4.28 e a função de ativação sigmóide logística definida na Equação 4.25.

A primeira derivada parcial do erro para obter o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$, pode ser obtido com a regra da cadeia da seguinte forma:

$$\frac{\partial E}{\partial \mathbf{w}_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial \mathbf{w}_j} + \frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2}\right)}{\partial \mathbf{w}_j}$$

A solução do primeiro fator do lado direito é descrita como:

$$\frac{\partial E}{\partial y_j} = \frac{\partial \left(\frac{1}{2} \sum_{c=1}^{C} (y_c - d_c)^2\right)}{\partial y_j}$$

$$= y_j - d_j.$$

Em seguida, o segundo fator pode ser definido como visto na Seção 4.6.3:

$$\frac{\partial y_j}{\partial z_j} = y_j \left(1 - y_j\right),$$

E, finalmente, o terceiro fator é

$$\frac{\partial z_j}{\partial \mathbf{w}_j} = \frac{\partial (\mathbf{x} \cdot \mathbf{w}_j)}{\partial \mathbf{w}_j} = \mathbf{x}$$

O gradiente do termo de regularização é

$$\frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2}\right)}{\partial \mathbf{w}_j} = \alpha \mathbf{w}_j$$

Então, a solução para o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$ é definida como segue:

$$\frac{\partial E}{\partial \mathbf{w}_j} = \left(y_j^2 - d_j \cdot y_j\right) \cdot (1 - y_j) \cdot \mathbf{x} + \alpha \mathbf{w}_j. \tag{4.33}$$

Para o método de Newton-Raphson, também deve ser obtida a segunda derivada parcial $\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}$, onde tem-se

$$\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k} = \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} \cdot \frac{\partial y_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial \mathbf{w}_k} + \frac{\partial \left(\alpha \mathbf{w}_j\right)}{\partial \mathbf{w}_k}$$

O primeiro fator do lado direito, para $j = k$, é definido como:

$$\begin{aligned}\frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} &= \frac{\partial\left(\left(y_j^2 - d_j \cdot y_j\right)\cdot(1-y_j)\cdot\mathbf{x}+\alpha\mathbf{w}_j\right)}{\partial y_k} \\ &= \left(-3y_j^2 + (2d_j+2)\,y_j - d_j\right)\cdot\mathbf{x}.\end{aligned}$$

Para $j \neq k$, tem-se

$$\frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} = 0.$$

O segundo fator torna-se

$$\frac{\partial y_k}{\partial z_k} = y_j\left(1 - y_j\right).$$

O terceiro fator é

$$\frac{\partial z_k}{\partial \mathbf{w}_k} = \mathbf{x}.$$

E, finalmente, para $j = k$ o termo de regularização é

$$\frac{\partial \alpha\mathbf{w}_j}{\partial \mathbf{w}_k} = \alpha,$$

em seguida, para $j \neq k$

$$\frac{\partial \alpha\mathbf{w}_j}{\partial \mathbf{w}_k} = 0.$$

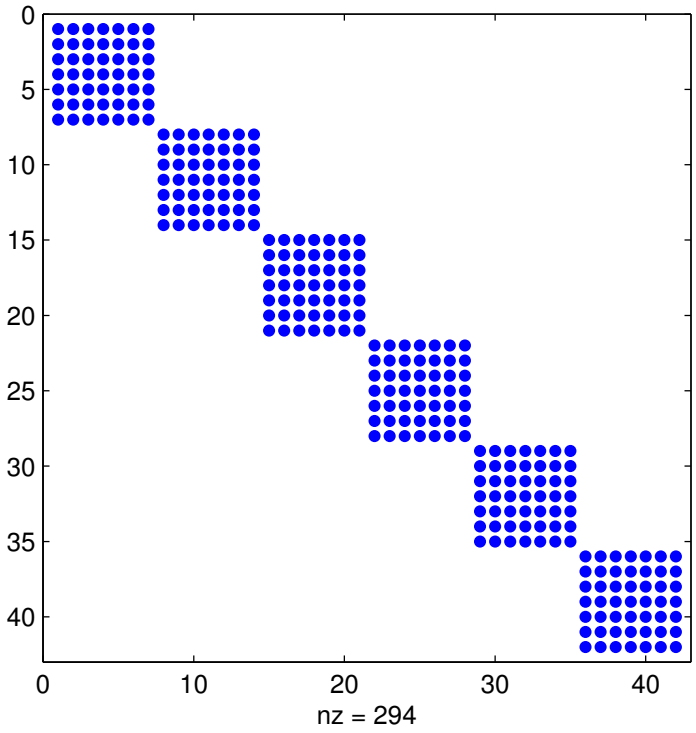
Então, como visto na Seção 4.5.4, a matriz hessiana é uma matriz bloco, onde cada bloco é

$$H_{jk} = \left[\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}\right] = \begin{cases} \sum_{n=1}^{N}\left\{\left(-3y_{nk}^2 + (2d_{nk}+2)\,y_{nk} - d_{nk}\right)\right. & \\ \left.\cdot\left(y_{nk} - y_{nk}^2\right)\cdot\mathbf{x}_n^T\mathbf{x}_n + \alpha\right\}, & \text{se } j = k \\ 0, & \text{se } j \neq k, \end{cases} \tag{4.34}$$

onde N é o número de instâncias do conjunto de dados.

Na Figura 15, podemos observar o preenchimento da matriz hessiana gerada pelo par de função de erro SSE e função de ativação sigmóide logística.

Figura 15 – Preenchimento da matriz hessiana gerada pelo par de função de erro SSE e função de ativação sigmóide logística, base de dados com seis classes e rede neural com seis neurônios na camada escondida.



4.6.7 Entropia Cruzada para Duas Classes e *Softmax*

Dada a função de erro entropia cruzada para duas classes definida a seguir.

$$E = -\sum_{c=1}^{C} d_c \cdot ln\left(y_c\right) + \left(1 - d_c\right) \cdot ln\left(1 - y_c\right), \tag{4.35}$$

onde C representa o número de classes ou nós de saída, y_c representa a saída obtida no nó c e d_c representa a saída desejada no respectivo nó.

Sendo a função de ativação *softmax* definida na Equação 4.19.

A primeira derivada parcial do erro para obter o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$, pode ser obtida com a regra da cadeia da seguinte forma:

$$\frac{\partial E}{\partial \mathbf{w}_j} = \left(\sum_{i=1}^{C} \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_j} \right) \cdot \frac{\partial z_j}{\partial \mathbf{w}_j} + \frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2} \right)}{\partial \mathbf{w}_j}$$

A solução do primeiro fator do lado direito é definida como:

$$\begin{aligned} \frac{\partial E}{\partial y_i} &= \frac{\partial \left(-\sum_{c=1}^{C} d_c \cdot ln(y_c) + (1-d_c) \cdot ln(1-y_c) \right)}{\partial y_i} \\ &= -\left(\frac{d_i}{y_i} - \frac{1-d_i}{1-y_i} \right). \end{aligned}$$

Em seguida, o segundo fator pode ser descrito como visto na Seção 4.6.1

$$\frac{\partial y_i}{\partial z_j} = y_i(\delta_{ij} - y_j),$$

onde $\delta_{ij} = \begin{cases} 1, & \text{se } i = j \\ 0, & \text{se } i \neq j \end{cases}$ é o delta de Kronecker.

E, finalmente, o terceiro fator é

$$\frac{\partial z_j}{\partial \mathbf{w}_j} = \frac{\partial (\mathbf{x} \cdot \mathbf{w}_j)}{\partial \mathbf{w}_j} = \mathbf{x}$$

O gradiente do termo de regularização é

$$\frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2} \right)}{\partial \mathbf{w}_j} = \alpha \mathbf{w}_j$$

Então, a solução para o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$ é definida como segue:

$$\frac{\partial E}{\partial \mathbf{w}_j} = -\sum_{i=1}^{C} \left[\left(d_i - \frac{y_i - d_i \cdot y_i}{1 - y_i} \right) \cdot (\delta_{ij} - y_j) \right] \cdot \mathbf{x} + \alpha \mathbf{w}_j. \tag{4.36}$$

Para o método de Newton-Raphson, também deve ser determinada a segunda derivada parcial $\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}$, onde tem-se

$$\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k} = \left(\sum_{l=1}^{C} \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_l} \cdot \frac{\partial y_l}{\partial z_k} \right) \cdot \frac{\partial z_k}{\partial \mathbf{w}_k} + \frac{\partial (\alpha \mathbf{w}_j)}{\partial \mathbf{w}_k}$$

O primeiro fator do lado direito, para $l = j$, é descrito como:

$$\frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_l} = \frac{\partial\left(-\sum_{i=1}^{C}\left[\left(d_i - \frac{y_i - d_i \cdot y_i}{1-y_i}\right)\cdot(\delta_{ij} - y_j)\right]\cdot\mathbf{x} + \alpha\mathbf{w}_j\right)}{\partial y_l}$$

$$= \left(\frac{1-d_l}{1-y_l}\right)\cdot\mathbf{x}.$$

Para $l \neq j$, tem-se

$$\frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_l} = \left(\frac{y_j - d_l \cdot y_j}{(1-y_l)^2}\right)\cdot\mathbf{x}.$$

O segundo fator torna-se

$$\frac{\partial y_l}{\partial z_k} = y_l(\delta_{lk} - y_k).$$

O terceiro fator é

$$\frac{\partial z_k}{\partial \mathbf{w}_k} = \mathbf{x}.$$

E, finalmente, para $j = k$ o termo de regularização é

$$\frac{\partial \alpha\mathbf{w}_j}{\partial \mathbf{w}_k} = \alpha,$$

em seguida, para $j \neq k$

$$\frac{\partial \alpha\mathbf{w}_j}{\partial \mathbf{w}_k} = 0.$$

Logo, $\frac{\partial \alpha\mathbf{w}_j}{\partial \mathbf{w}_k}$ pode ser reescrito como visto na Seção 4.6.1:

$$\frac{\partial \alpha\mathbf{w}_j}{\partial \mathbf{w}_k} = \alpha \cdot \delta_{jk}.$$

Então, como visto na Seção 4.5.4, a matriz hessiana é uma matriz bloco, onde cada bloco

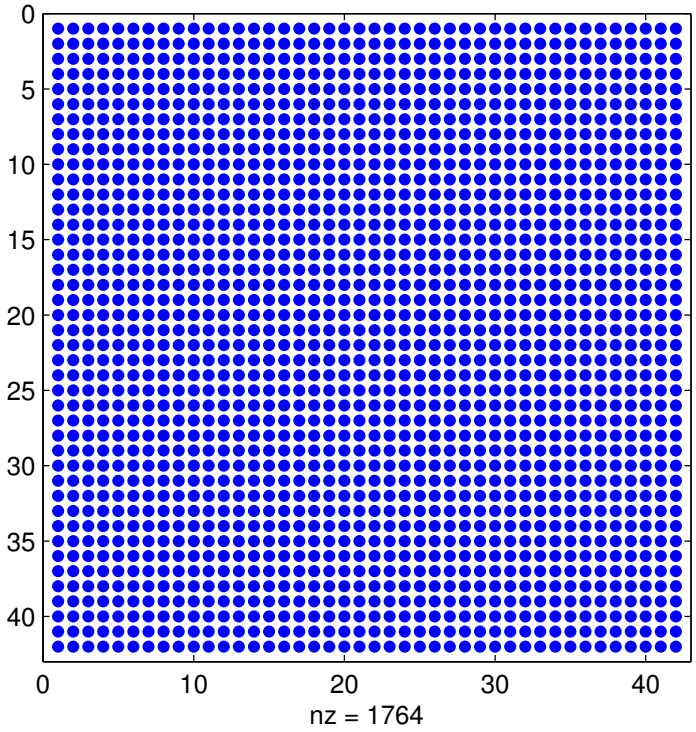
é

$$
\begin{aligned}
H_{jk} &= \left[\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}\right] \\
&= \sum_{n=1}^{N} \left\{ \left[\sum_{l=1}^{C} \begin{cases} \left(\frac{1-d_l}{1-y_l}\right) \cdot y_{nl} \cdot (\delta_{lk} - y_{nk}), & \text{se } l = j \\ \left(\frac{y_j - d_l \cdot y_j}{(1-y_l)^2}\right) \cdot y_{nl} \cdot (\delta_{lk} - y_{nk}), & \text{se } l \neq j \end{cases} \right] \right. \\
&\quad \left. \cdot \mathbf{x}_n^T \mathbf{x}_n + \alpha \cdot \delta_{jk} \right\},
\end{aligned} \tag{4.37}
$$

onde N é o número de instâncias do conjunto de dados.

Na Figura 16, podemos observar o preenchimento da matriz hessiana gerada pelo par de função de erro entropia cruzada para duas classes e função de ativação *softmax*.

Figura 16 – Preenchimento da matriz hessiana gerada pelo par de função de erro entropia cruzada para duas classes e função de ativação *softmax*, base de dados com seis classes e rede neural com seis neurônios na camada escondida.



4.6.8 Entropia Cruzada para Duas Classes e Linear

Sendo a função de erro entropia cruzada para duas classes definida na Equação 4.35 e a função de ativação linear como definida na Equação 4.22.

A primeira derivada parcial do erro para obter o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$, pode ser obtido com a regra da cadeia da seguinte forma:

$$\frac{\partial E}{\partial \mathbf{w}_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial \mathbf{w}_j} + \frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2}\right)}{\partial \mathbf{w}_j}$$

A solução do primeiro fator do lado direito é definida como:

$$\begin{aligned} \frac{\partial E}{\partial y_j} &= \frac{\partial \left(-\sum_{c=1}^{C} d_c \cdot ln(y_c) + (1-d_c) \cdot ln(1-y_c)\right)}{\partial y_j} \\ &= -\left(\frac{d_j}{y_j} - \frac{1-d_j}{1-y_j}\right). \end{aligned}$$

Em seguida, o segundo fator pode ser definido como:

$$\frac{\partial y_j}{\partial z_j} = 1,$$

E, finalmente, o terceiro fator é

$$\frac{\partial z_j}{\partial \mathbf{w}_j} = \frac{\partial (\mathbf{x} \cdot \mathbf{w}_j)}{\partial \mathbf{w}_j} = \mathbf{x}$$

O gradiente do termo de regularização é

$$\frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2}\right)}{\partial \mathbf{w}_j} = \alpha \mathbf{w}_j$$

Então, a solução para o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$ é definida como segue:

$$\frac{\partial E}{\partial \mathbf{w}_j} = -\left(\frac{d_j}{y_j} - \frac{1-d_j}{1-y_j}\right) \cdot \mathbf{x} + \alpha \mathbf{w}_j. \tag{4.38}$$

Para o método de Newton-Raphson, também deve ser obtida a segunda derivada parcial $\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}$, onde tem-se

$$\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k} = \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} \cdot \frac{\partial y_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial \mathbf{w}_k} + \frac{\partial \left(\alpha \mathbf{w}_j\right)}{\partial \mathbf{w}_k}$$

O primeiro fator do lado direito, para $j = k$, é descrito como:

$$\begin{aligned} \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} &= \frac{\partial\left(-\left(\frac{d_j}{y_j} - \frac{1-d_j}{1-y_j}\right) \cdot \mathbf{x} + \alpha \mathbf{w}_j\right)}{\partial y_k} \\ &= \left(\frac{d_j}{y_j^2} + \frac{1-d_j}{(1-y_j)^2}\right) \cdot \mathbf{x}. \end{aligned}$$

Para $j \neq k$, tem-se

$$\frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} = 0.$$

O segundo fator torna-se

$$\frac{\partial y_k}{\partial z_k} = 1.$$

O terceiro fator é

$$\frac{\partial z_k}{\partial \mathbf{w}_k} = \mathbf{x}.$$

E, finalmente, para $j = k$ o termo de regularização é

$$\frac{\partial \alpha \mathbf{w}_j}{\partial \mathbf{w}_k} = \alpha,$$

em seguida, para $j \neq k$

$$\frac{\partial \alpha \mathbf{w}_j}{\partial \mathbf{w}_k} = 0.$$

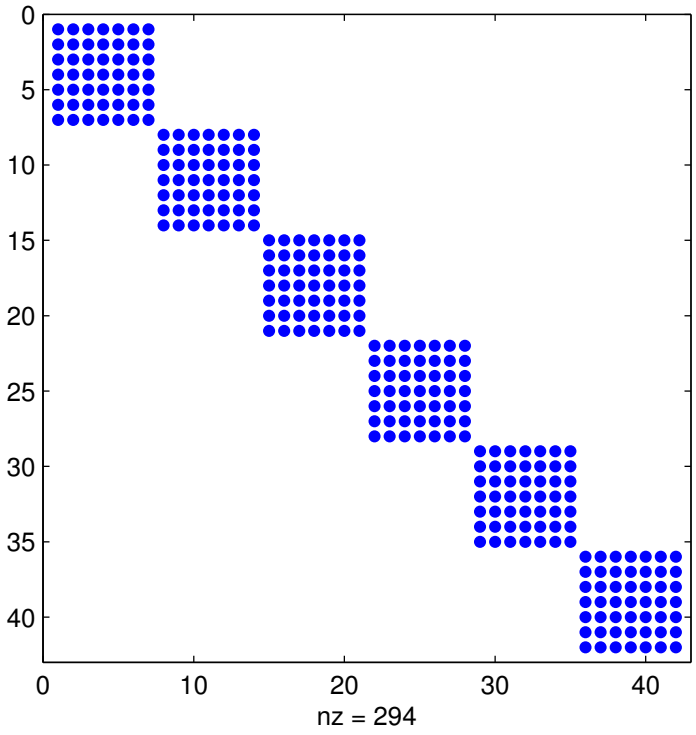
Então, como visto na Seção 4.5.4, a matriz hessiana é uma matriz bloco, onde cada bloco é

$$H_{jk} = \left[\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}\right] = \begin{cases} \sum_{n=1}^{N} \left\{\left(\frac{d_{nk}}{y_{nk}^2} + \frac{1-d_{nk}}{(1-y_{nk})^2}\right) \cdot \mathbf{x}_n^T \mathbf{x}_n + \alpha\right\}, & \text{se } j = k \\ 0, & \text{se } j \neq k, \end{cases} \tag{4.39}$$

onde N é o número de instâncias do conjunto de dados.

Na Figura 17, podemos observar o preenchimento da matriz hessiana gerada pelo par de função de erro entropia cruzada para duas classes e função de ativação linear.

Figura 17 – Preenchimento da matriz hessiana gerada pelo par de função de erro entropia cruzada para duas classes e função de ativação linear, base de dados com seis classes e rede neural com seis neurônios na camada escondida.



4.6.9 Entropia Cruzada para Duas Classes e Sigmóide Logística

Dada a função de erro entropia cruzada para duas classes definida na Equação 4.35 e a função de ativação sigmóide logística definida na Equação 4.25.

A primeira derivada parcial do erro para obter o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$, pode ser obtido com a regra da cadeia da seguinte forma:

$$\frac{\partial E}{\partial \mathbf{w}_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial \mathbf{w}_j} + \frac{\partial \left(\alpha \frac{\mathbf{w}^T \mathbf{w}}{2} \right)}{\partial \mathbf{w}_j}$$

A solução do primeiro fator do lado direito é definida como:

$$\frac{\partial E}{\partial y_j} = \frac{\partial\left(-\sum_{c=1}^{C} d_c \cdot ln(y_c)+(1-d_c)\cdot ln(1-y_c)\right)}{\partial y_j}$$

$$= -\left(\frac{d_j}{y_j} - \frac{1-d_j}{1-y_j}\right).$$

Em seguida, o segundo fator pode ser determinado como visto na Seção 4.6.3:

$$\frac{\partial y_j}{\partial z_j} = y_j\left(1 - y_j\right),$$

E, finalmente, o terceiro fator é

$$\frac{\partial z_j}{\partial \mathbf{w}_j} = \frac{\partial(\mathbf{x} \cdot \mathbf{w}_j)}{\partial \mathbf{w}_j} = \mathbf{x}$$

O gradiente do termo de regularização é

$$\frac{\partial\left(\alpha\frac{\mathbf{w}^T\mathbf{w}}{2}\right)}{\partial \mathbf{w}_j} = \alpha\mathbf{w}_j$$

Então, a solução para o gradiente $\frac{\partial E}{\partial \mathbf{w}_j}$ é definida como segue:

$$\frac{\partial E}{\partial \mathbf{w}_j} = (y_j - d_j) \cdot \mathbf{x} + \alpha\mathbf{w}_j. \tag{4.40}$$

Para o método de Newton-Raphson, também deve ser obtida a segunda derivada parcial $\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}$, onde tem-se

$$\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k} = \frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} \cdot \frac{\partial y_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial \mathbf{w}_k} + \frac{\partial\left(\alpha\mathbf{w}_j\right)}{\partial \mathbf{w}_k}$$

O primeiro fator do lado direito, para $j = k$, é definido como:

$$\frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} = \frac{\partial((y_j - d_j)\cdot\mathbf{x} + \alpha\mathbf{w}_j)}{\partial y_k}$$

$$= \mathbf{x}.$$

Para $j \neq k$, tem-se

$$\frac{\frac{\partial E}{\partial \mathbf{w}_j}}{\partial y_k} = 0.$$

O segundo fator torna-se

$$\frac{\partial y_k}{\partial z_k} = y_j \left(1 - y_j\right).$$

O terceiro fator é

$$\frac{\partial z_k}{\partial \mathbf{w}_k} = \mathbf{x}.$$

E, finalmente, para $j = k$ o termo de regularização de peso é

$$\frac{\partial \alpha \mathbf{w}_j}{\partial \mathbf{w}_k} = \alpha,$$

em seguida, para $j \neq k$

$$\frac{\partial \alpha \mathbf{w}_j}{\partial \mathbf{w}_k} = 0.$$

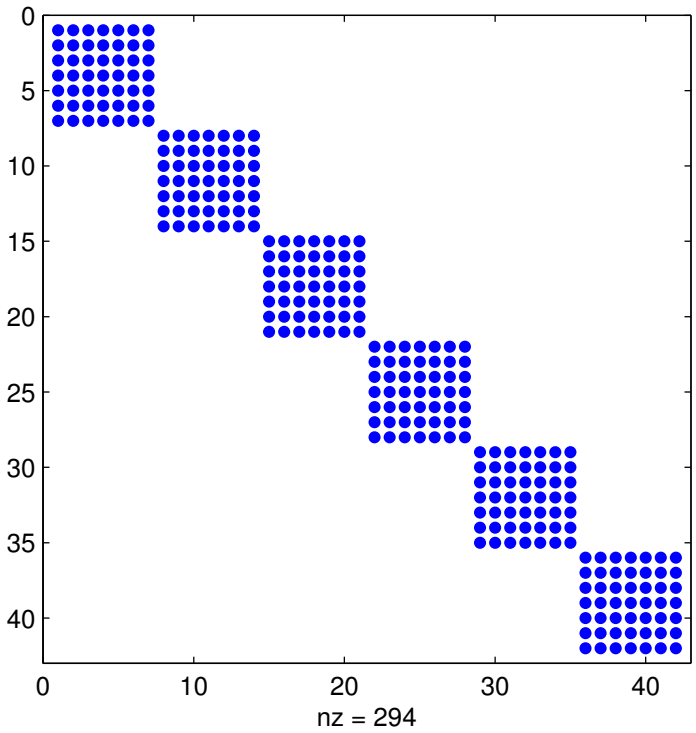
Então, como visto na Seção 4.5.4, a matriz hessiana é uma matriz bloco, onde cada bloco é

$$H_{jk} = \left[\frac{\partial^2 E}{\partial \mathbf{w}_j \partial \mathbf{w}_k}\right] = \begin{cases} \sum_{n=1}^{N} \left\{(y_{nk} - y_{nk}^2) \cdot \mathbf{x}_n^T \mathbf{x}_n + \alpha\right\}, & \text{se } j = k \\ 0, & \text{se } j \neq k, \end{cases} \tag{4.41}$$

onde N é o número de instâncias do conjunto de dados.

Na Figura 18, podemos observar o preenchimento da matriz hessiana gerada pelo par de função de erro entropia cruzada para duas classes e função de ativação sigmóide logística.

Figura 18 – Preenchimento da matriz hessiana gerada pelo par de função de erro entropia cruzada para duas classes e função de ativação sigmóide logística, base de dados com seis classes e rede neural com seis neurônios na camada escondida.



4.7 Conclusões

Neste capítulo, comtemplamos as RNAs e suas arquiteturas. Definimos funções de ativação e funções de erro. Discutimos sobre as redes RBF, sua arquitetura e seu treinamento. Mostramos o algoritmo IRLS para treinamento de redes neurais e as derivações necessárias para a utilização deste algoritmo. Enfim, discutimos sobre o termo de regularização como uma forma de controlar a complexidade da rede.

No próximo capítulo, definiremos os pares naturais de funções de erro e funções de ativação e mostraremos os pares naturais e não naturais utilizados neste trabalho.

5 Pares Naturais

A função de ativação e a função de erro são fundamentais para a aprendizagem da rede. A função de ativação é reponsável por definir, com o vetor de entrada ponderado pelo vetor de pesos, a saída de cada neurônio da rede. Já a função de erro calcula o quão distante a saída da rede está da saída desejada. Sendo assim, as funções de ativação e as de erro são partes importantes na tarefa de minimização do erro, e por sua vez na tarefa de aprendizagem das redes neurais.

Neste capítulo, definimos pares naturais (assim nomeados por Bishop [2006]) de funções de ativação e funções de erro e mostraremos os pares naturais e não naturais utilizados neste trabalho.

5.1 Definição

Dado $\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial z_j}$ e o gradiente da função de erro em relação ao vetor de pesos $\frac{\partial E}{\partial \mathbf{w}_j} = \frac{\partial E}{\partial z_j} \cdot \frac{\partial z_j}{\partial \mathbf{w}_j}$, onde Δ_j mede o erro do neurônio j, como visto na Seção 4.5.3. Segundo Bishop [2006], Dunne and Campbell [1997], um par de função de erro e ativação é dito par natural se $\frac{\partial E}{\partial z_j}$ assume a seguinte forma:

$$\frac{\partial E}{\partial z_j} = y_j - d_j,$$

onde a medida do erro do neurônio j é a diferença entre a saída obtida e a saída desejada do respectivo neurônio.

Na Tabela 1, podemos verificar os valores que $\frac{\partial E}{\partial z_j}$ assume para cada par de funções utilizados neste trabalho. De acordo com a definição dada nesta seção, os pares SSE e linear, entropia cruzada para duas classes e sigmóide logística, entropia Cruzada Multiclasses e *sotfmax* podem ser ditos pares naturais.

Tabela 1 – $\frac{\partial E}{\partial z_j}$ para os pares de funções de erro e funções de ativação

	SSE	**Entropia Cruzada para Duas Classes**	**Entropia Cruzada Multiclasses**
Linear	$y_j - d_j$	$\frac{1-d_j}{1-y_j} - \frac{d_j}{y_j}$	$-\frac{d_j}{y_j}$
Sigmóide Log.	$\left(y_j^2 - d_j \cdot y_j\right) \cdot (1 - y_j)$	$y_j - d_j$	$d_j \cdot y_j - d_j$
Sotfmax	$\sum_{i=1}^{C} \left[(y_i^2 - d_i \cdot y_i) \cdot(\delta_{ij} - y_j)\right]$	$-\sum_{i=1}^{C} \left[\left(d_i - \frac{y_i - d_i \cdot y_i}{1-y_i}\right) \cdot(\delta_{ij} - y_j)\right]$	$y_j - d_j$

5.2 Conclusões

Neste capítulo, definimos pares naturais de funções de erro e funções de ativação e mostramos $\frac{\partial E}{\partial z_j}$ para cada par de funções utilizados neste trabalho.

No próximo capítulo, mostraremos as bases de dados utilizadas nos experimentos, como foram realizados os ajustes de parâmetros das redes neurais e os resultados obtidos.

6 Experimentos

Experimentos são feitos com o intuito de comprovar ou verificar, na prática, uma ou mais hipóteses. Neste trabalho, utilizamos a plataforma MATLAB [MathWorks, 1970] para desenvolver e executar os experimentos, também utilizamos a biblioteca Netlab [Nabney and Bishop, 2003].

Neste capítulo, apresentaremos as bases de dados utilizadas nos experimentos, os passos realizados para encontrar o conjunto de parâmentros para cada rede neural que permitiram uma melhor generalização, e os resultados obtidos.

6.1 Bases de Dados

Nesta seção, descreveremos as bases de dados utilizadas, assim como, a dimensionalidade dos dados, o número de instâncias e o número de classes de cada base de dados.

6.1.1 Bases de Dados Artificiais

Com o intuito de análisar as fronteiras de decisão[1] de cada rede neural artificial com diferentes pares de funções de erro e ativação, mostraremos, nesta seção, três bases de dados artificiais: Duas Meias-Luas, Duas Meias-Luas Invertidas e Seis Gaussianas.

Na base de dados Duas Meias-Luas, encontramos 326 instâncias com 2 dimensões e 2 classes. Na Figura 19, podemos ver a disposição dos dados num gráfico de 2 dimensões. Já na base de dados Duas Meias-Luas Invertidas existem 460 instâncias com duas dimensões e 2 classes. A distribuição dos dados da base Duas Meias-Luas Invertidas é mostrada na Figura 20.

[1] Fronteira de Decisão é uma hipersuperfice gerada por um algoritmo de aprendizagem de máquina que particiona o espaço vetorial em vários conjuntos, um para cada classe.

Figura 19 – Base de dados Duas Meias-Luas.

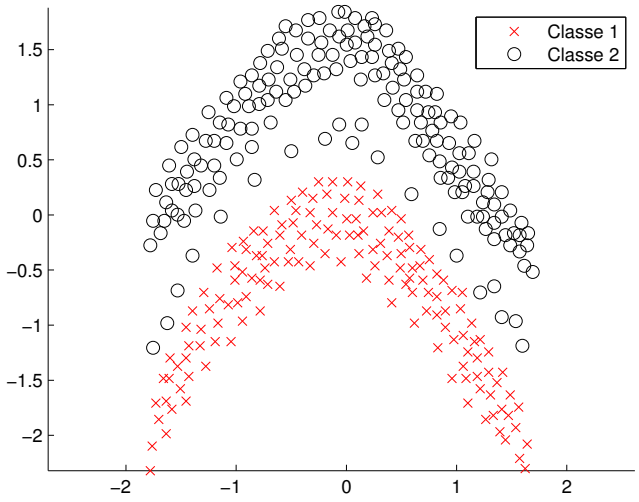
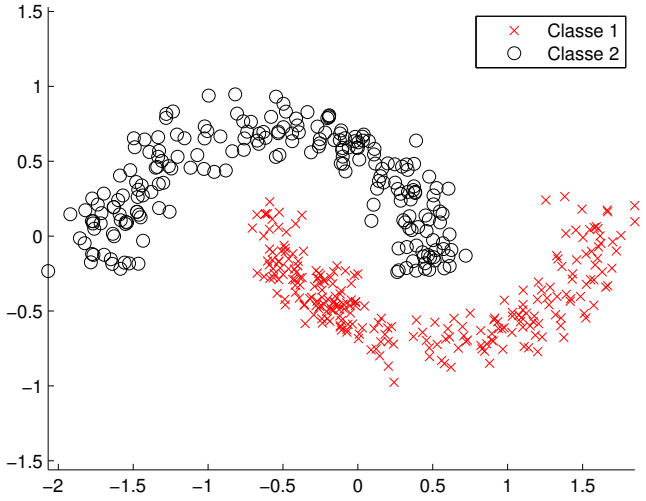
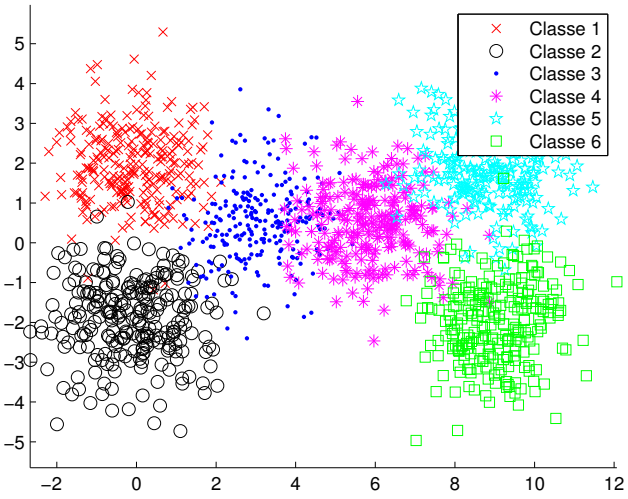


Figura 20 – Base de dados Duas Meias-Luas Invertidas.



Com 1500 instâncias, a base de dados Seis Gaussianas contém dados com 2 dimensões e 6 classes. Na Figura 21, podemos ver a disposição das instâncias da base Seis Gaussianas.

Figura 21 – Base de dados Seis Gaussianas.



Enfim, mostramos na Tabela 2 um resumo das informações mostradas nesta seção.

Tabela 2 – Bases de dados artificiais, número de instâncias, dimensionalidade dos dados e quantidade de classes.

Base de Dados	**Número de Instâncias**	**Atributos ou Dimensões**	**Quantidade de Classes**
Duas Meias-Luas	326	2	2
Duas Meias-Luas Invertidas	460	2	2
Seis Gaussianas	1500	2	6

6.1.2 Bases de Dados do Mundo Real

Usamos também 5 bases do mundo real: *Pen-Based Recognition of Handwritten Digits*, *Optical Recognition of Handwritten Digits*, *Semeion Handwritten Digit*, *Letter Image Recognition Data* e *Landsat Satellite Data Set*. Todos os dados foram obtidos em [Asuncion and Newman, 2007].

A primeira base de dados, *Pen-Based Recognition of Handwritten Digits*, consiste de 10992 amostras de digitos escritos à mão, utilizando um *WACOM PL-100V pressure sensitive tablet*, obtidas de 44 escritores. A base de dados foi posteriormente reduzida, através de uma amostragem estratificada, para 1099 instâncias. Ela tem 10 classes, representando cada dígitos, e 16 dimensões. Segundo autor, a dimensionalidade das instâncias foi obtida através de uma interpolação linear das imagens com 500×500 *pixels*.

Em *Optical Recognition of Handwritten Digits*, encontramos 5620 amostras de digitos manuscritos em um formulário pré-processado. Um total de 43 pessoas contribuiram para a obtenção dos dados contidos na base, segundo o autor. Através de uma amostragem estratificada, a base de dados foi reduzida para 1033 instâncias. Cada amostra da base de dados representa uma imagem 32×32 *pixels* com os valores de cada *pixel* discretizados em $\{0, 1\}$ (preto ou branco). Cada imagem foi dividida em blocos de 4×4 *pixels* pelos autores gerando uma matriz 8×8 com valores em um intervalo inteiro de $[0, 16]$, onde cada valor da matriz representa a soma dos valores de *pixel* de cada bloco. A base de dados contém 10 classes, onde cada classe representa um dígito.

Com 1593 amostras de dígitos manuscritos, *Semeion Handwritten Digit* é um conjunto de dados onde cada amostra representa uma imagem com 16×16 *pixels* em uma escala de cinza de 256 valores. Cada *pixel* de cada amostra foi discretizado em $\{0, 1\}$ (preto ou branco), pelos autores da base de dados, usando um limiar fixo. A quantidade de instâncias do conjunto de dados foi reduzida, através de uma amostragem estratificada, para 716 instâncias. A base de dados tem uma classe para cada dígito, resultando em um total de 10 classes.

Contendo imagens preto e branco de caracteres do alfabeto inglês, *Letter Image Recognition Data* é um conjunto de dados de imagens de caracteres de 20 fontes diferentes com 20000 amostras. Cada amostra foi transformada em 16 atributos (dimensões) numéricos primitivos e posteriormente escalonados num intervalo inteiro $[0, 15]$. O conjunto de dados tem 26 classes representando cada letra do alfabeto inglês e foi reduzido estratificadamente para 1012 instâncias.

A base de dados *Landsat Satellite Data Set* consiste de imagens multiespectrais[2] 3×3 da vizinhança. Com dados de 36 dimensões (4 bandas espectrais $\times$ 9 *pixels*), o objetivo consiste em classificar as amostras em 7 classes diferentes: solo vermelho, cultura de algodão, solo cinza, solo cinza úmido, solo cinza muito úmido, solo com vegetação restolho ou classe mistura (todos os tipos de presentes). A base de dados consiste em 6435 instâncias, reduzida estratificadamente para 1020 instâncias. Cada atributo está em $[0, 255]$.

Finalmente, na Tabela 3 podemos visualizar as bases de dados vistas nesta seção e seu repectivo número de instâncias, seu número de atributos ou dimensões e sua quantidade

[2] Uma imagem multiespectral consiste em imagens de um mesmo objeto, obtidas com diferentes comprimentos de ondas eletromagnéticas. Tendo como exemplo a luz visível, infravermelha, ultravioleta, raio-X ou qualquer outra faixa do espectro.

de classes.

Tabela 3 – Bases de dados, número de instâncias, dimensionalidade dos dados e quantidade de classes.

Base de Dados	**Número de Instâncias**	**Atributos ou Dimensões**	**Quantidade de Classes**
Pen-Based Recognition of Handwritten Digits	1099	16	10
Optical Recognition of Handwritten Digits	1033	64	10
Semeion Handwritten Digit	716	256	10
Letter Image Recognition Data	1012	16	26
Landsat Satellite Data Set	1020	36	7

6.2 Ajustes de Parâmetros

O ajuste de parâmetros tem o objetivo de identificar o conjunto de parâmetros que oferece o melhor desempenho na resolução de um problema. Nesta seção mostraremos as variações de parâmetros e o método utilizado para a escolha do melhor conjunto de parâmetros. Todos os valores que os parâmetros assumiram foram escolhido de forma empírica.

6.2.1 Bases de Dados Artificiais

Com o intuito de obter parâmetros que levam a rede neural a alcançar uma fronteira de decisão ótima, variamos os parâmetros da seguinte maneira:

- Coeficiente do termo de regularização $\alpha \in \{10^{-1}, 1, 10, 10^2, 10^3\}$;
- Largura da Função Radial
  $\sigma \in \{10^{-1}, 2 \cdot 10^{-1}, 3 \cdot 10^{-1}, 4 \cdot 10^{-1}, 5 \cdot 10^{-1}, 6 \cdot 10^{-1}, 7 \cdot 10^{-1}, 8 \cdot 10^{-1}, 9 \cdot 10^{-1}, 1\}$;
- Número de iterações $t \in \{10, 50, 100, 500, 1000\}$;
- Número de neurônios escondidos $W = 100$.

Selecionamos as melhores fronteiras de decisão através de inspeção visual baseando-se em sua suavidade e generalização.

6.2.2 Bases de Dados do Mundo Real

Para tentar obter melhor generalização, ajustamos os parâmetros das RBFs da seguinte maneira:

- Coeficiente do termo de regularização $\alpha \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4\}$;
- Largura da Função Radial $\sigma \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$;
- Número de iterações $t \in \{25, 100, 500, 1000\}$;
- Número de neurônios escondidos $W = 100$.

Usamos validação cruzada [Kohavi et al., 1995][3,4] de *k-folds* para selecionar os parâmetros potencialmente ideais para as RBFs. Utilizando $k = 10$, obtivemos, para cada base de dados, a média do desempenho de predição dos dados de validação para a escolha do conjunto de parâmetros ideal. Para realizar a comparação da capacidade de generalização de cada RBF, utilizamos a média e desvio padrão do desempenho dos dados de teste gerados pela rede neural com o conjunto de parâmetros de melhor desempenho de validação.

É importante ressaltar que utilizamos uma normalização em $[0, 1]$ na saída da rede neural para calcular a função de erro nos experimentos das redes neurais com as funções entropia cruzada e linear, entropia cruzada para duas classes e linear, a fim de evitar valores de saída negativos, tendo em vista o cálculo das funções logarítmicas na função de erro.

Nas Tabelas 4 a 8, podemos observar os parâmetros com melhores desempenhos de validação em relação as diferentes bases de dados.

[3] O método de validação cruzada *k-fold* que utilizamos neste trabalho consiste em dividir o conjunto total de dados em k subconjuntos mutuamente exclusivos de mesmo tamanho. O processo de predição é realizado k vezes alternando o subconjunto de treinamento, validação e teste. Ao final são produzidas k predições.

[4] Neste trabalho, dividimos, de forma estratificada, os conjuntos de dados em conjuntos de treinamento, validação e teste com 80%, 10% e 10% dos dados, repectivamente.

Tabela 4 – Melhores parâmetros para a base de dados *Pen-Based Recognition of Handwritten Digits*.

	Pen-Based Recognition of Handwritten Digits			
	α	σ	t	W
Entropia Cruzada e Sigmóide Logítica	100	1	25	100
Entropia Cruzada e Linear	1000	1	1000	100
Entropia Cruzada e *Softmax*	10	1	1000	100
Entropia Cruzada para Duas Classes e Sigmóide Logítica	10	1	100	100
Entropia Cruzada para Duas Classes e Linear	0,001	0,1	1000	100
Entropia Cruzada para Duas Classes e *Softmax*	1	0,1	25	100
SSE e Sigmóide Logítica	100	1	100	100
SSE e Linear	0,001	1	1000	100
SSE e *Softmax*	0,01	1	1000	100

Tabela 5 – Melhores parâmetros para a base de dados *Optical Recognition of Handwritten Digits*.

	Optical Recognition of Handwritten Digits			
	α	σ	t	W
Entropia Cruzada e Sigmóide Logítica	100	1	100	100
Entropia Cruzada e Linear	0,001	1	25	100
Entropia Cruzada e *Softmax*	10	1	500	100
Entropia Cruzada para Duas Classes e Sigmóide Logítica	10	1	500	100
Entropia Cruzada para Duas Classes e Linear	0,001	1	1000	100
Entropia Cruzada para Duas Classes e *Softmax*	1000	1	500	100
SSE e Sigmóide Logítica	10	1	500	100
SSE e Linear	0,001	1	100	100
SSE e *Softmax*	10	1	1000	100

Tabela 6 – Melhores parâmetros para a base de dados *Semeion Handwritten Digit*.

	Semeion Handwritten Digit			
	α	σ	t	W
Entropia Cruzada e Sigmóide Logítica	0,001	1	100	100
Entropia Cruzada e Linear	100	1	25	100
Entropia Cruzada e *Softmax*	10	1	100	100
Entropia Cruzada para Duas Classes e Sigmóide Logítica	1	1	1000	100
Entropia Cruzada para Duas Classes e Linear	0,001	1	1000	100
Entropia Cruzada para Duas Classes e *Softmax*	10	1	1000	100
SSE e Sigmóide Logítica	10	1	1000	100
SSE e Linear	0,1	1	100	100
SSE e *Softmax*	1	1	1000	100

Tabela 7 – Melhores parâmetros para a base de dados *Letter Image Recognition Data*.

	Letter Image Recognition Data			
	α	σ	t	W
Entropia Cruzada e Sigmóide Logítica	100	1	1000	100
Entropia Cruzada e Linear	0,01	10	1000	100
Entropia Cruzada e *Softmax*	1	1	500	100
Entropia Cruzada para Duas Classes e Sigmóide Logítica	1	1	1000	100
Entropia Cruzada para Duas Classes e Linear	0,001	0,1	500	100
Entropia Cruzada para Duas Classes e *Softmax*	10	1	500	100
SSE e Sigmóide Logítica	10	1	1000	100
SSE e Linear	0,001	1	500	100
SSE e *Softmax*	0,1	1	100	100

Tabela 8 – Melhores parâmetros para a base de dados *Landsat Satellite Data Set*.

	Landsat Satellite Data Set			
	α	σ	t	W
Entropia Cruzada e Sigmóide Logítica	10	1	100	100
Entropia Cruzada e Linear	1000	1	100	100
Entropia Cruzada e *Softmax*	10	1	1000	100
Entropia Cruzada para Duas Classes e Sigmóide Logítica	1	1	1000	100
Entropia Cruzada para Duas Classes e Linear	0,001	0,1	1000	100
Entropia Cruzada para Duas Classes e *Softmax*	1000	1	1000	100
SSE e Sigmóide Logítica	100	1	500	100
SSE e Linear	0,001	1	500	100
SSE e *Softmax*	100	1	25	100

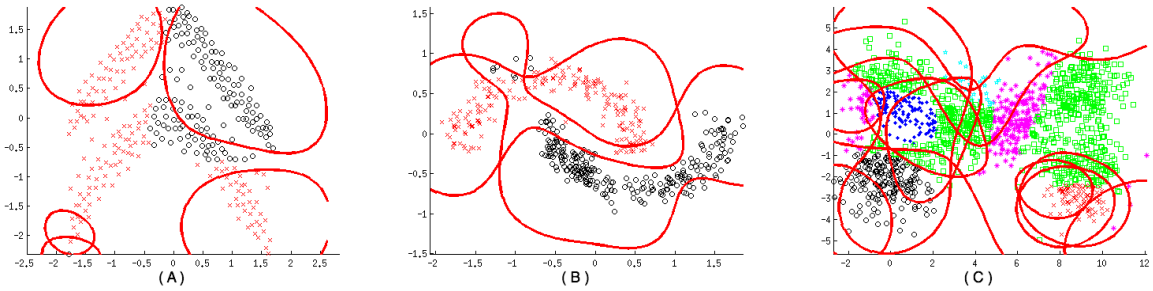
6.3 Resultados

Nesta seção, apresentaremos os desempenhos de generalização de cada RBF com as bases de dados do mundo real, bem como gráficos que demonstram as fronteiras de decisão geradas para as bases de dados artificiais.

6.3.1 Bases de Dados Artificiais

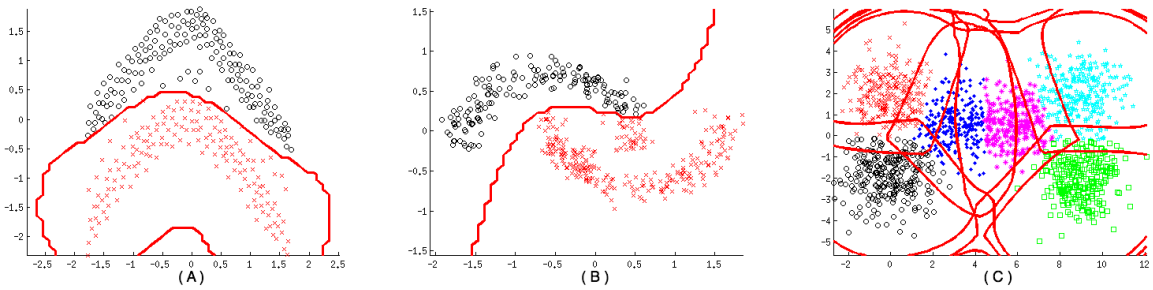
Podemos visualizar as fronteiras de decisão para cada rede neural com diferentes funções de ativação e funções de erro nas Figuras 22 a 30.

Figura 22 – Fronteira de decisão da rede neural com as funções entropia cruzada e linear para as bases de dados (A) duas meias-luas, (B) duas meias-luas invertidas e (C) seis gaussianas.



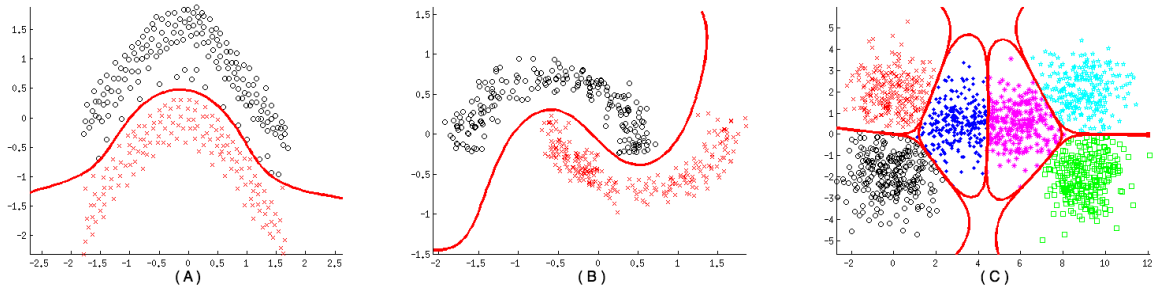
Podemos observar que, em sua maioria, as redes neurais com os diferentes pares de funções geraram boas fronteiras de decisão, obtendo uma boa generalização. Entretanto, as Figuras 22 e 25 mostram que as redes neurais com os pares entropia cruzada e linear, entropia cruzada para duas classes e linear não alcançaram um mínimo da função de erro e por isso não geraram boas fronteiras.

Figura 23 – Fronteira de decisão da rede neural com as funções entropia cruzada e sigmóide logística para as bases de dados (A) duas meias-luas, (B) duas meias-luas invertidas e (C) seis gaussianas.



Na Figura 23 (A e B) vemos que as redes neurais com o par de funções entropia cruzada e sigmóide logística produzem fronteiras de decisão ásperas. Já na Figura 23 (C) um conjunto de fronteiras sobrepostas é produzido, a fim de obter uma boa generalização.

Figura 24 – Fronteira de decisão da rede neural com as funções entropia cruzada e *softmax* para as bases de dados (A) duas meias-luas, (B) duas meias-luas invertidas e (C) seis gaussianas.



Podemos também destacar a rede neural com o par de funções entropia cruzada e *softmax* que obtiveram um ótimo grau de generalização, como podemos ver na Figura 24.

Figura 25 – Fronteira de decisão da rede neural com as funções entropia cruzada para duas classes e linear para as bases de dados (A) duas meias-luas, (B) duas meias-luas invertidas e (C) seis gaussianas.

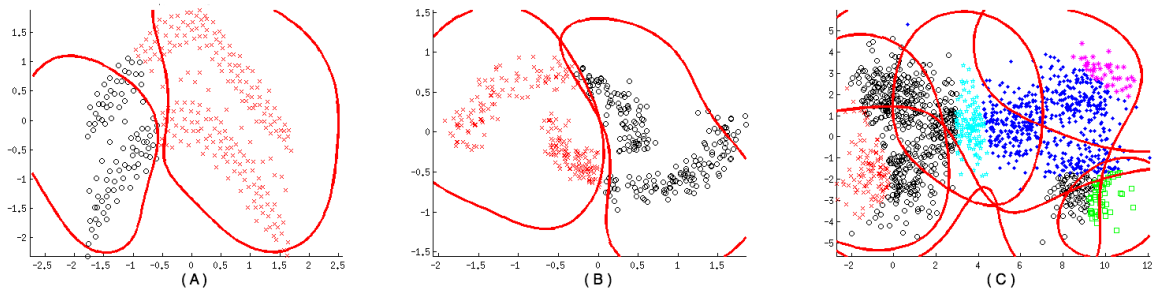


Figura 26 – Fronteira de decisão da rede neural com as funções entropia cruzada para duas classes e sigmóide logística para as bases de dados (A) duas meias-luas, (B) duas meias-luas invertidas e (C) seis gaussianas.

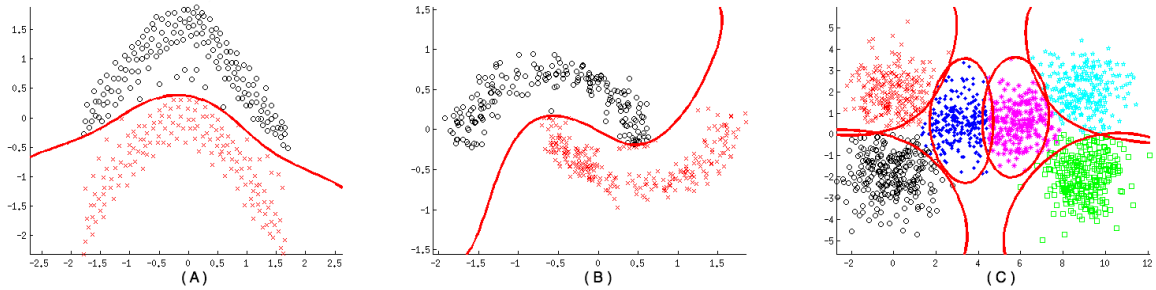
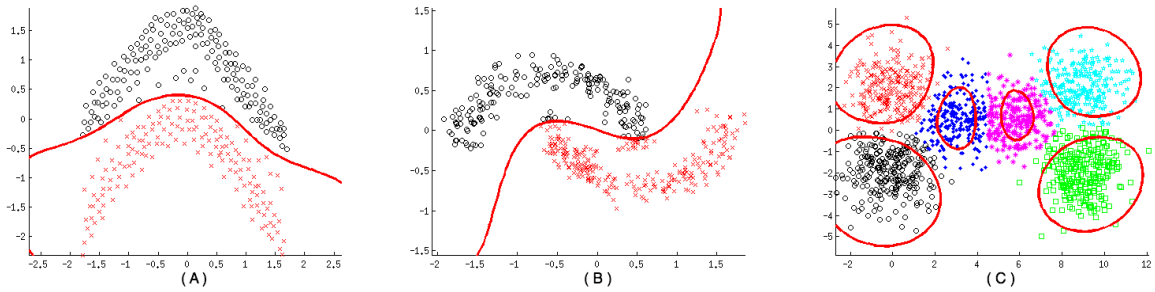


Figura 27 – Fronteira de decisão da rede neural com as funções entropia cruzada para duas classes e *softmax* para as bases de dados (A) duas meias-luas, (B) duas meias-luas invertidas e (C) seis gaussianas.



É possível observar nas Figuras 26 a 29 (A e B) que as redes neurais responsáveis por gerar as fronteiras de decisão vistas nas imagens citadas apresentam algo próximo da generalização ótima.

Figura 28 – Fronteira de decisão da rede neural com as funções SSE e sigmóide logística para as bases de dados (A) duas meias-luas, (B) duas meias-luas invertidas e (C) seis gaussianas.

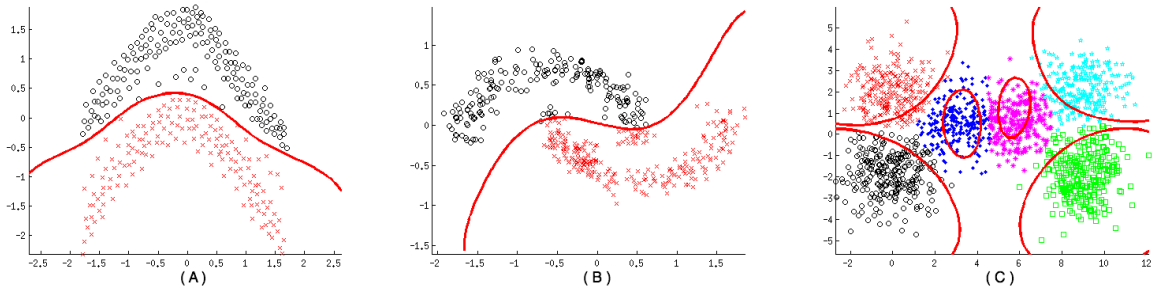


Figura 29 – Fronteira de decisão da rede neural com as funções SSE e *softmax* para as bases de dados (A) duas meias-luas, (B) duas meias-luas invertidas e (C) seis gaussianas.

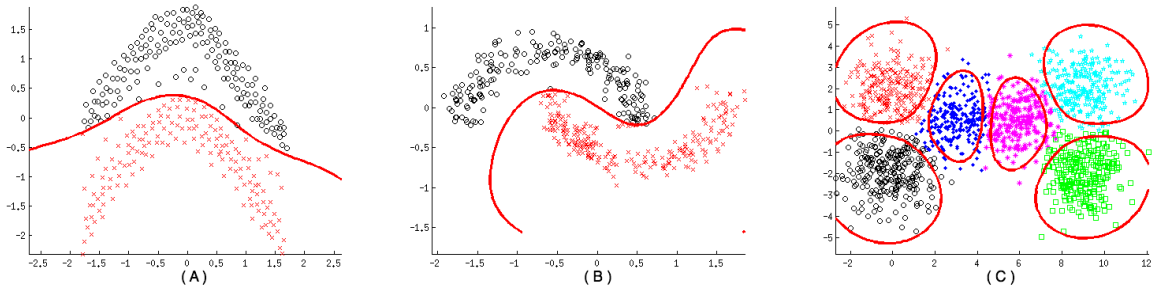
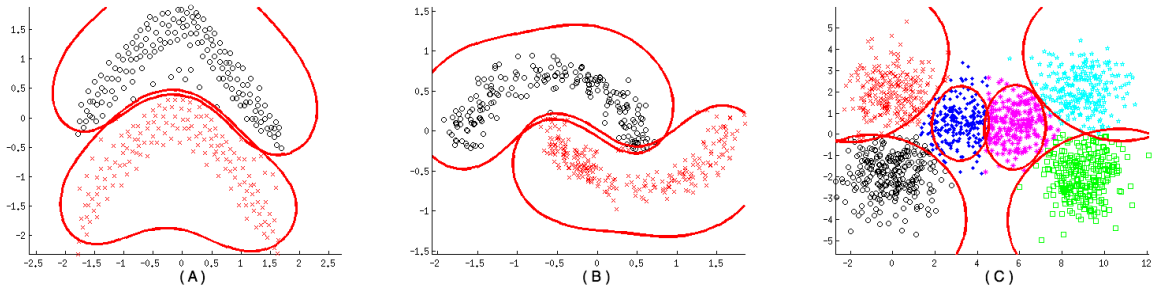


Figura 30 – Fronteira de decisão da rede neural com as funções SSE e linear para as bases de dados (A) duas meias-luas, (B) duas meias-luas invertidas e (C) seis gaussianas.



Enfim, a Figura 30 mostra que as redes neurais com o par de função SSE e linear gera uma fronteira de decisão para cada classe.

6.3.2 Bases de Dados do Mundo Real

Nas Tabelas 9 e 10, podemos verificar os resultados dos experimentos com as bases de dados do mundo real, onde a tradução das siglas contidas nas tabelas pode ser feita da seguinte maneira:

- Função de Erro Entropia Cruzada (EC);
- Função de Erro Entropia Cruzada para Duas Classes (ECD);
- Função de Erro Soma dos Quadrados do Erro (SSE);

- Função de Ativação *Softmax* (S);
- Função de Ativação Sigmóide Logística (SL);
- Função de Ativação Linear (L);
- Base de dados *Pen-Based Recognition of Handwritten Digits* (PD);
- Base de dados *Optical Recognition of Handwritten Digits* (OD);
- Base de dados *Semeion Handwritten Digit* (SD);
- Base de dados *Letter Image Recognition Data* (LET);
- Base de dados *Landsat Satellite Data Set* (SAT);

Tabela 9 – Resultados dos experimentos (Percentual de Acertos)

	PD	**OD**	**SD**	**SAT**	**LET**
	$\overline{x} \pm \sigma$	$\overline{x} \pm \sigma$	$\overline{x} \pm \sigma$	$\overline{x} \pm \sigma$	$\overline{x} \pm \sigma$
EC e SL	$31,28\% \pm 7,45\%$	$29,12\% \pm 13,97\%$	$15,21\% \pm 7,85\%$	$61,37\% \pm 6,2\%$	$7,52\% \pm 2,68\%$
EC e L	$15,59\% \pm 6,11\%$	$12,13\% \pm 3,83\%$	$10,98\% \pm 2,8\%$	$37,15\% \pm 12,55\%$	$3,96\% \pm 1,68\%$
EC e S	$76,14\% \pm 6,76\%$	$85,33\% \pm 3,92\%$	$57,04\% \pm 6,58\%$	$81,76\% \pm 3,03\%$	$56,93\% \pm 7,71\%$
ECD e SL	$74,49\% \pm 7,22\%$	$83,1\% \pm 5,08\%$	$73,09\% \pm 20,91\%$	$85,68\% \pm 3,03\%$	$55,05\% \pm 7,06\%$
ECD e L	$50,45\% \pm 5,18\%$	$37,18\% \pm 23,56\%$	$20,98\% \pm 14,73\%$	$52,74\% \pm 6,56\%$	$8,71\% \pm 3,26\%$
ECD e S	$45,87\% \pm 4,65\%$	$22,52\% \pm 9,38\%$	$18,45\% \pm 17,21\%$	$63,03\% \pm 5,27\%$	$17,82\% \pm 15,31\%$
SSE e SL	$31,28\% \pm 13,17\%$	$36,6\% \pm 24,51\%$	$18,31\% \pm 10,72\%$	$57,05\% \pm 22,69\%$	$8,71\% \pm 6,15\%$
SSE e L	$95,87\% \pm 2,08\%$	$95,53\% \pm 2,82\%$	$89,15\% \pm 5,51\%$	$88,84\% \pm 2,45\%$	$71,88\% \pm 6,55\%$
SSE e S	$30,18\% \pm 15,57\%$	$33,49\% \pm 21,83\%$	$38,02\% \pm 15,99\%$	$56,76\% \pm 16,27\%$	$16,43\% \pm 10,13\%$

Nota: Na tabela, $\overline{x}$ representa a média e σ o desvio padrão

Na Tabela 9, podemos verificar a média e o desvio padrão das redes neurais para cada base de dados.

Tabela 10 – Teste t de Student aplicado aos resultados.

	Vitórias	**Empates**	**Derrotas**
SSE e L	40	0	0
ECD e SL	32	3	5
EC e S	30	3	7
SSE e S	11	12	17
ECD e S	8	16	16
ECD e L	8	13	19
EC e SL	5	16	19
SSE e SL	4	18	18
EC e L	0	3	37

Na Tabela 10, verificamos o resultado da aplicação do teste de hipótese[5,6] teste t de Student[7] [Fisher, 1925] nos resultados da Tabela 9. Dado que o teste t verifica a hipótese em relação à média de duas amostras, utilizamos o método de vitória, empate e derrota para avaliar o desempenho de cada rede neural com os diferentes pares de funções em relação a cada base de dados. Dadas duas médias $\overline{x}_1$ e $\overline{x}_2$, uma vitória é declarada para $\overline{x}_1$ quando a hipótese nula for rejeitada e $\overline{x}_1 > \overline{x}_2$, caso $\overline{x}_1 < \overline{x}_2$ é declarado uma derrota do algoritmo de média $\overline{x}_1$. Um empate é declarado quando a hipótese nula for aceita.

Também na Tabela 10, podemos ver que o par de funções com melhor resultado foi a função de erro SSE e a função de ativação linear (L). O pior resultado foi dado pelo par de função de erro entropia cruzada e função ativação linear (EC eL).

6.4 Conclusões

Neste capítulo, apresentamos as bases de dados utilizadas nos experimentos, onde dedicamos três bases de dados artificiais à análise de fronteiras de decisão e cinco base de dados do mundo real à comparação experimental da capacidade de generalização dos algoritmos. Mostramos também como foi realizado os ajustes de parâmetros. Utilizamos o método de validação cruzada para definir os melhores conjuntos de parâmetros para as bases de dados do mundo real. Finalmente, reportamos os resultados obtidos para as bases de dados artificiais e as bases de dados do mundo real. Em nossos experimentos, pudemos notar que o melhor par de função de erro e ativação foi o par função de erro SSE e função de ativação linear. No próximo capítulo, discutiremos os resultados obtidos e suas possíveis causas.

[5] Teste de Hipótese é o procedimento ou regra de decisão que nos possibilita decidir por aceitar a hipótese nula ou não, com base na informação contida na amostra, onde a hipótese nula é a hipótese que se toma como verdadeira.

[6] Neste trabalho, consideramos a igualdade entre os resultados de duas redes neurais, em relação a uma base de dados, a hipótese nula.

[7] O teste t de Student é o teste de hipótese que compara duas médias baseado na distribuição de probabilidade t de Student.

7 Análise dos Resultados

Podemos definir uma análise como um exame detalhado daquilo que se procura analisar. Neste capítulo analisaremos os resultados mostrados no Capítulo 6.

7.1 Discussões

Na Tabela 10, com relação às funções de erro, podemos notar que a função de erro SSE obteve um total de 55 vitórias nas 3 funções de ativação. Por sua vez, com relação às funções de ativação, temos que a função *softmax* obteve um total de 49 vitórias nas 3 funções de erro.

Apesar de muitos pares não naturais terem gerado boas fronteiras de decisão com as bases artificiais, é incontestável, de acordo com a Tabela 10, que os pares naturais de funções de ativação tiveram os melhores desempenhos nos experimentos realizados com as bases do mundo real, estando todos entre os três melhores pares.

Nos dados artificiais, observarmos que o par de funções entropia cruzada e *softmax* obtiveram uma fronteira de decisão muito bem ajustada e bem generalizada, tendo assim uma das melhores fronteiras de decisão. Entretanto, nos experimentos das bases do mundo real, teve o pior resultado entre os pares naturais de funções de erro e de ativação, ficando abaixo até do par entropia cruzada para duas classes e sigmóide logística. Isso pode ter sido causado por um *overfitting*, contudo, não podemos descartar a possibilidade de ter sido causado pelo fato de que os problemas de reconhecimento de imagens tratados neste trabalho podem ser relativamente simples, em contrapartida, a rede com o par entropia cruzada e *softmax* pode ter propriedades com complexidade além do necessário para a resolução dos problemas, dando assim, lugar para as redes com o par de função SSE e linear obterem o melhor desempenho.

É possível observar que, de acordo com a Tabela 9, alguns pares de funções tiveram desvios padrão relativamente altos em diferentes bases de dados. Também podemos observar, na mesma tabela, que alguns pares de funções tem média de acertos de predição, em diferentes bases, abaixo de 50%. Tudo isso pode ter sido causado por mínimos locais na superfície de erro, pontos de sela ou pontos de máximo, tendo em vista que o sentido de Newton

aponta para um mínimo, não sendo necessariamente um mínimo global, podendo também apontar para um ponto de sela ou até mesmo para um ponto de máximo, como visto na Seção 4.5.4.

Analisando os pares de função de erro e ativação entropia cruzada e linear, entropia cruzada para duas classes e linear, podemos observar que os dois pares de funções obtiveram desempenho baixo, onde o par entropia cruzada e linear teve os piores resultados de acordo com as Tabelas 9 e 10. A causa desse problema pode ter sido em razão da natureza probabilistica das funções de erro entropia cruzada, algo que não está inerente na função de ativação linear. Tendo em vista também que a função de ativação sigmóide logística é um caso especial da função *softmax*[Dunne and Campbell, 1997].

Em relação as bases de dados artificiais, vemos que, em sua maioria, as redes neurais alcançaram um bom grau de generalização. Já nas bases de dados do mundo real, muitas das redes neurais utilizadas obtiveram resultados de baixa qualidade.

Na Tabela 9, vemos que, para a base de dados *Letter Image Recognition Data*, as redes neurais alcançaram um baixo desempenho, chegando até $96,04\%$ de erro com um desvio padrão de $1,68\%$. Entretando, exite a possibilidade da base de dados ter uma alta sobreposições entre classes, levando os dados a serem de difícil aprendizagem. Também existe a possibilidade de que o pré-processamento aplicado à base de dados citada não tenha sido o mais apropriado para destacar as propriedades de cada imagem.

Por fim, nos resultados relacionados às bases de dados *Pen-Based Recognition of Handwritten Digits*, *Optical Recognition of Handwritten Digits*, *Semeion Handwritten Digit* e *Landsat Satellite Data Set* podemos identificar, de acordo com a Tabela 9, resultados de baixa qualidade. Porém, existem resultados, como os obtidos pela rede neural com o par de funções SSE e linear, que nos mostram que as bases de dados citadas não têm muitos dados sobrepostos.

7.2 Conclusões

Neste capítulo, discutimos os resultados, vistos nas Tabelas 9 e 10, e suas possíveis causas. Vimos que de fato os pares naturais tiveram melhores desempenhos na resolução dos problemas de reconhecimento de imagens tratados neste trabalho. Por fim, observamos que existe a possibilidade de uma das cinco bases de dados utilizadas neste trabalho ter

uma alta sobreposições entre classes.

8 Conclusão

Com o objetivo de realizar experimentos com bases de dados de reconhecimento de imagens para apontar o impacto de pares naturais de funções de erro e funções de ativação em redes neurais, neste trabalho, mostramos como são representadas as imagens digitais e os passos para o processamento de imagens. Apresentamos como são definidos os neurônios artificiais, as redes neurais e suas arquiteturas. Contemplamos a definição de funções de ativações e funções de erro em redes neurais, as redes RBF utilizadas neste trabalho e o algoritmo IRLS utilizado para treinamento das redes neurais.

Devido à utilização de um método de segunda ordem para treinamento das redes neurais, mostramos as derivadas parciais necessárias na aplicação do algoritmo de treinamento utilizado. Definimos pares naturais de funções de erro e funções de ativação, mostramos os pares utilizados neste trabalho e a diferença entre os pares naturais e não naturais.

No Capítulo 6, descrevemos as bases de dados utilizadas, que por problemas de desempenho foram reduzidas estratificadamente. Também mostramos como foram feitos os ajustes de parâmetros das redes neurais e os resultados obtidos. Finalizando este trabalho, no Capítulo 7 discutimos os resultados obtidos.

Por fim, utilizando redes RBF e o algoritmo de treinamento IRLS, observamos que as redes neurais com pares naturais de funções de erro e funções de ativação mostraram resultados superiores às redes com pares não naturais quando aplicadas em alguns problemas de reconhecimento de imagens.

Como trabalhos futuros podemos incluir:

- Aplicar a comparação experimental a outras bases de dados;
- A utilização de outras redes neurais, como as redes MLP;
- A utilização de outros algoritmos para o treinamento das redes neurais;
- Estudar outros problemas envolvendo redes neurais artificiais ou outros algoritmos de aprendizagem de máquina.

Referências

A. Asuncion and D. Newman. Uci machine learning repository. `http://archive.ics.uci.edu/ml/`, 2007.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

C. M. Bishop et al. Neural networks for pattern recognition. 1995.

D. S. Broomhead and D. Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document, 1988.

M. D. Buhmann. *Radial basis functions: theory and implementations*, volume 12. Cambridge university press, 2003.

O. Chapelle, B. Schölkopf, A. Zien, et al. Semi-supervised learning. 2006.

A. de Pádua Braga, A. P. de L. F. de Carvalho, and T. B. Ludermir. *Redes Neurais Artificiais: Teoria e Aplicações*. LTC Editora, Rio de Janeiro-RJ, 2000.

R. A. Dunne and N. A. Campbell. On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function. In *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne, 181*, volume 185. Citeseer, 1997.

S. E. Fahlman. An empirical study of learning speed in back-propagation networks. 1988.

R. A. Fisher. *Statistical methods for research workers*. Genesis Publishing Pvt Ltd, 1925.

K. Fukumizu and S.-i. Amari. Local minima and plateaus in multilayer neural networks. 1999.

R. C. Gonzalez and R. E. Woods. *Processamento de Imagens Digitais*. Editora Edgard Blucher, São Paulo-SP, 2000.

M. Gori and A. Tesi. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86, 1992.

D. M. Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

S. Haykin. *Redes Neurais: Princípios e Prática*. Bookman Companhia Editora, Porto Alegre-RS, 2001.

S. Hayman. The mcculloch-pitts model. In *Neural Networks, 1999. IJCNN'99. International Joint Conference on*, volume 6, pages 4438–4439. IEEE, 1999.

J. Hertz, A. Krogh, R. G. Palmer, and H. Horner. Introduction to the theory of neural computation. *Physics Today*, 44(12):70–70, 2008.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.

Y. Le Cun, O. Matan, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jacket, and H. Baird. Handwritten zip code recognition with multilayer networks. In *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, volume 2, pages 35–40. IEEE, 1990.

J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

E. Masson and Y.-J. Wang. Introduction to computation and learning in artificial neural networks. *European Journal of Operational Research*, 47(1):1–28, 1990.

MathWorks. Matlab®. `http://www.mathworks.com/products/matlab/`, 1970.

J. L. McClelland, D. E. Rumelhart, P. R. Group, et al. Parallel distributed processing. *Explorations in the microstructure of cognition*, 2:216–271, 1986.

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

J. M. Mendel and R. W. McLaren. Reinforcement-learning control and pattern recognition systems. In *Adaptive, learning, and pattern recognition systems; theory and applications*, pages 287–318. Academic Press, 1970.

I. Nabney and C. Bishop. Netlab neural network software. `http://www.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/`, 2003.

R. Neruda and P. Kudová. Learning methods for radial basis function networks. *Future Generation Computer Systems*, 21(7):1131–1142, 2005.

G. G. Parma, B. R. D. MENEZES, and A. P. Braga. Neural networks learning with sliding mode control: the sliding mode backpropagation algorithm. *International Journal of Neural Systems*, 9(03):187–193, 1999.

M. Riedmiller and H. Braun. Rprop-a fast adaptive learning algorithm. In *Proc. of ISCIS VII), Universitat*. Citeseer, 1992.

F. Rosenblatt. The perceptron–a perceiving and recognizing automaton. Technical report, Cornell Aeronautical Laboratory, 1957.

H. Rowley, S. Baluja, T. Kanade, et al. Neural network-based face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1):23–38, 1998.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5, 1988.

M. Sonka, V. Hlavac, and R. Boyle. *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

B. A. Warner and M. Misra. Iteratively reweighted least squares based learning. In *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*, volume 2, pages 1327–1331. IEEE, 1998.

P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

B. WIDROW, M. E. HOFF, et al. Adaptive switching circuits. 1960.