ESTUDO COMPARATIVO DE TÉCNICAS DE MAPEAMENTO NO CLASSIFICADOR WISARD

Gabriel Pereira Guarisa

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Priscila Machado Vieira Lima
Felipe Maia Galvão França

Rio de Janeiro
Dezembro de 2020

ESTUDO COMPARATIVO DE TÉCNICAS DE MAPEAMENTO NO CLASSIFICADOR WISARD

Gabriel Pereira Guarisa

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientadores: Priscila Machado Vieira Lima
Felipe Maia Galvão França

Aprovada por: Prof. Priscila Machado Vieira Lima
Prof. Geraldo Bonorino Xexéo
Prof. Alexandre Solon Nery

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2020

Guarisa, Gabriel Pereira

Estudo comparativo de técnicas de mapeamento no classificador WiSARD/Gabriel Pereira Guarisa. – Rio de Janeiro: UFRJ/COPPE, 2020.

XII, 59 p.: il.; 29, 7cm.

Orientadores: Priscila Machado Vieira Lima

Felipe Maia Galvão França

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2020.

Referências Bibliográficas: p. 39 – 43.

1. WiSARD. 2. Mapeamento. 3. Otimização de hiperparâmetros. 4. Redes neurais sem peso. I. Lima, Priscila Machado Vieira *et al.*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

A Deus, pela minha vida, e por todas as oportunidades e superações.

Aos meus pais Alene e Claudio, e meu irmão Davi, que me incentivaram e me aturaram nos momentos de maior impaciência.

Aos meus professores, em especial aos meus orientadores Felipe e Priscila, que com paciência e dedicação me guiaram ao longo da realização deste trabalho.

A todos os meus amigos, obrigado pela compreensão nos meus muitos momentos de ausência. Aos novos amigos que fiz durante o curso, em destaque a sociedade composta por Aluizio, Leopoldo e Luiz, por todos os períodos de falta de esperança compartilhados. Aos meus amigos da *Blue House*, pelas risadas e conversas sem sentido.

A todos aqueles que contribuíram, de alguma forma, para a realização deste trabalho.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ESTUDO COMPARATIVO DE TÉCNICAS DE MAPEAMENTO NO CLASSIFICADOR WISARD

Gabriel Pereira Guarisa

Dezembro/2020

Orientadores: Priscila Machado Vieira Lima
Felipe Maia Galvão França

Programa: Engenharia de Sistemas e Computação

A otimização de hiper-parâmetros é uma questão de grande relevância na área de aprendizado de máquina, uma vez que sua escolha é fundamental para o procedimento de treinamento, resultando na otimização de quesitos como a velocidade e acurácia dos algoritmos. O modelo WiSARD é um classificador de ênuplas com múltiplos discriminadores, sendo conhecido pela velocidade ao usar o processo de escrita nas RAMs como treinamento. Tal modelo é composto por três diferentes hiper-parâmetros, que dizem respeito ao tamanho da ênupla, o pré-processamento das entradas, e o mapeamento das entradas, que define a divisão em ênuplas, fator determinante na acurácia do classificador. Um estudo comparativo das técnicas de mapeamento para o modelo WiSARD é apresentado neste trabalho. Para tal, diferentes abordagens de mapeamento presente na literatura do modelo são descritas, e novas técnicas são apresentadas.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

COMPARATIVE STUDY OF MAPPING TECHNIQUES IN THE WISARD CLASSIFIER

Gabriel Pereira Guarisa

December/2020

Advisors: Priscila Machado Vieira Lima
Felipe Maia Galvão França

Department: Systems Engineering and Computer Science

The optimization of hyperparameters is a matter of great relevance in the area of machine learning, since its choice is fundamental for the training procedure, resulting in the optimization of issues such as the speed and accuracy of the algorithms. The WiSARD model is a tuple classifier with multiple discriminators and is known for its speed when it uses the RAM writing process as training. Such a model consists of three different hyperparameters, which concern the size of the tuple, the preprocessing of the entries, and the mapping of the entries, which defines the division into tuples, a determining factor in the accuracy of the classifier. A comparative study of the mapping techniques for the WiSARD model is presented in this work. To this end, different approaches to mapping present in the model literature are described, and new techniques are presented.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Na área de aprendizado de máquinas, os modelos são compostos por parâmetros aprendidos durante o treinamento, e hiper-parâmetros, que controlam o processo do aprendizado. O ajuste dos hiper-parâmetros é considerado um problema de otimização, onde a escolha de tais valores pode afetar a velocidade e a qualidade do treinamento [1].

Em [2], o Wilkes, Stonham *and* Aleksander *Recognition Device* (WiSARD) é apresentado como um modelo que utiliza memórias de acesso aleatório para armazenar o conhecimento aprendido e, através de padrões binários divididos em ênuplas, os procedimentos de escrita e leitura são equivalentes ao treinamento e classificação, respectivamente.

Além do procedimento de binarização e do tamanho das ênuplas no modelo WiSARD, o mapeamento que define a divisão dos padrões em ênuplas também é um hiper-parâmetro, e sua escolha é decisiva no aprendizado [3]. Entretanto, apesar de sua importância, a exploração do espaço de todos os mapeamentos é inviável devido ao número expressivo de combinações possíveis, de forma que, por padrão estabelecido na literatura, sua escolha é feita de forma aleatória [4].

1.1 Motivação

A otimização de hiper-parâmetros é um passo essencial na aplicação prática de determinados algoritmos de aprendizado de máquina, uma vez que o tempo de treinamento torna a escolha *ad-hoc* dos valores de hiper-parâmetros proibitiva [5]. Todavia, quando comparado com redes neurais com pesos sinápticos, o tempo de treinamento no modelo WiSARD é significativamente menor, tornando pouco atrativo o desenvolvimento de novas tecnologias voltadas para sua redução.

Para mais, na última década, o procedimento foi responsável pelo avanço do estado-da-arte em problemas de classificação de imagens [6]. Tal característica foi pouco explorada na literatura do modelo WiSARD, com uma escassez de modelos

para a criação de mapeamentos utilizando o conhecimento presente nos conjuntos de treinamento, uma vez que os mapeamentos tem impacto direto na acurácia do classificador.

Além do número reduzido de métodos para a criação dos mapeamentos de ênuplas, há pouca diversidade nas abordagens existentes. Consequentemente, não existem estudos que comparem as técnicas de criação de mapeamentos no modelo.

1.2 Objetivos e contribuições

O presente trabalho tem como objetivo principal definir e analisar diferentes técnicas de criação de mapeamentos para o modelo WiSARD a partir do conhecimento presente nos conjuntos de treinamento, que resultem em uma melhor acurácia do classificador. Para a realização de tal objetivo, algumas metas foram estipuladas, a saber:

- Apresentação e uso dos métodos já existentes;
- Elaboração de novas técnicas para criação dos mapeamentos;
- Análise dos resultados das abordagens aplicadas em conjuntos de dados relevantes.

Desta forma, a principal contribuição do trabalho realizado é apresentar um estudo comparativo de abordagens para a criação de mapeamentos de ênuplas no modelo WiSARD.

1.3 Estrutura da dissertação

O Capítulo 2 apresenta o conceito de rede neural sem peso, dando foco no modelo WiSARD, relatando o histórico do modelo, sua estrutura e trabalhos relacionados. O Capítulo 3 introduz os conceitos base aos modelos desenvolvidos no presente trabalho, e descreve as técnicas presentes na literatura para a criação de mapeamentos. No Capítulo 4 são descritas as estratégias e os modelos desenvolvidos para a criação dos mapeamentos a partir dos conjuntos de dados. O Capítulo 5 aponta os *datasets* utilizados e expõe os resultados das abordagens quando comparadas com as abordagens padrões da literatura, apresentando as devidas análises dos resultados encontrados. Por fim, o Capítulo 6 conclui destacando os pontos alcançados e apontando possíveis trabalhos futuros. O trabalho ainda conta com o Apêndice A, que contém as tabelas de resultados que detalham as configurações dos resultados experimentais. O Apêndice B traz os artigos desenvolvidos em co-autoria durante a confecção da presente dissetação.

Capítulo 2

O Modelo WiSARD de redes neurais sem pesos

Neste capítulo são apresentados os conceitos teóricos relacionados a redes neurais artificiais sem pesos, partindo do advento do classificador de ênuplas e, posteriormente, o modelo WiSARD, com seu funcionamento, particularidades e extensões sendo descritas. Por fim, são listadas as estratégias de criação de mapeamentos presentes na literatura.

2.1 O classificador de ênuplas

Introduzido no ano de 1959, o classificador de ênuplas é um método de reconhecimento de padrões, inicialmente elaborado para um equipamento de identificação de dígitos e caracteres [7]. Seu funcionamento se dá através de matrizes de memória indexadas por ênuplas compostas de *pixels* de entradas binárias apresentadas através da "retina"do dispositivo. No caso de um mapeamento exclusivo, quando não há repetição de endereços, o tamanho da retina é igual a m vezes n, onde m é a quantidade de matrizes de memória do classificador.

A escrita na matriz de memória, denominada de etapa de aprendizado, é a alteração de 0 para 1 no *bit* de acesso armazenado no endereço determinado pela ênupla e pela classe da imagem. A matriz de memória é composta por linhas que representam todas as 2^n combinações possíveis dos bits das ênuplas, enquanto as colunas da matriz indicam qual a classe do padrão aprendido (ver Figura 2.1).

No processo de reconhecimento, o padrão apresentado é dividido em ênuplas utilizando o mesmo mapeamento de endereços da etapa de aprendizado. No entanto, o conteúdo da matriz de memória é apenas lido e, uma vez que a classe é desconhecida, os valores de todas as classes para aquele determinado endereço são utilizados. O procedimento de leitura é realizado em todas as matrizes de memória, o que resulta

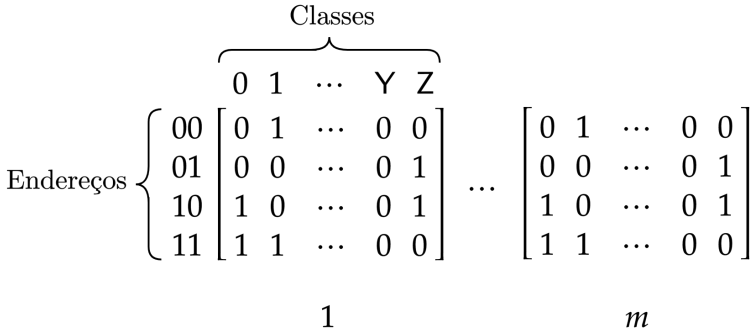


Figura 2.1: Exemplo de matrizes de memória para dígitos e caracteres com n igual a 2.

em uma matriz $\mathbf{R}$, onde as linhas são as respostas das matrizes de memória e as colunas as classes. A classificação é realizada através do somatório das linhas para cada coluna da matriz $\mathbf{R}$, a resposta y' do classificador será a classe que possuir o maior valor (ver Equação 2.1).

$$y' = argmax_c \sum_{i=1}^{i \leq m} R_{i,c} \tag{2.1}$$

Uma modificação no processo de escrita nas matrizes de memória é introduzida em [8], deixando de utilizar os indicadores booleanos e adotando contadores que indicam a frequência que um determinado endereço foi acessado na etapa de aprendizado. Tal mudança possibilitou melhores resultado ao reduzir os empates durante a classificação.

2.2 WiSARD: arquitetura básica e treinamento

Utilizando a divisão da entrada em ênuplas, o modelo WiSARD (acrônimo para Wilkes, Stonhan and Aleksander's Recognition Device) tem como principal diferencial a utilização de RAMs (do inglês *Random Access Memory*) no lugar das matrizes de memória. Diferentemente da abordagem de matrizes, cada classe possui um conjunto próprio de m RAMs, chamado de discriminador (ver Figura 2.2).

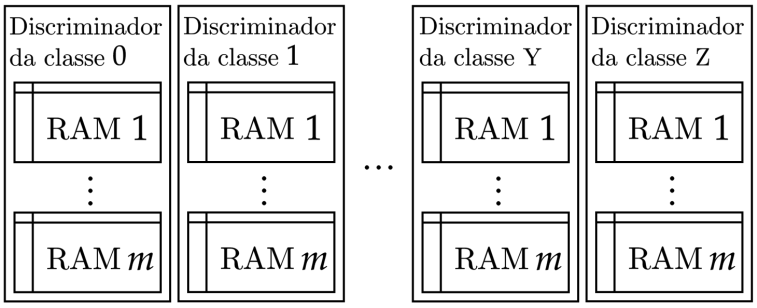


Figura 2.2: Exemplo da estrutura de discriminadores do modelo WiSARD.

Cada RAM é eficiente em reconhecer um padrão apresentado anteriormente, mas não é capaz de generalizar a informação, só reconhecendo padrões vistos [2]. A generalização ocorre através da combinação do conhecimento de diferentes RAMs [9] e não há uso de técnicas de minimização de erros [10] . Tal característica torna o custo computacional de treinamento do modelo igual ao da escrita nas RAMs.

Junto com o advento dos discriminadores surgiu a possibilidade de mapeamentos independentes, diferentemente do que era utilizado com apenas um mapeamento compartilhado entre todas as classes, tornando o uso de um mesmo mapeamento opcional.

Em um primeiro momento, as variáveis presentes nos endereços de memória no modelo WiSARD também eram booleanas. Entretanto, tal abordagem leva a uma saturação das RAMs, o que prejudica, e até mesmo inviabiliza, o reconhecimento de padrões. A solução encontrada, de forma semelhante ao que foi feito no classificador de ênuplas, foi utilizar um inteiro que é incrementado toda vez que a posição de memória é visitada durante o treinamento [11].

2.3 WiSARD: classificação

O modelo WiSARD é considerado uma rede neural sem pesos, em contraparte a redes neurais artificiais que utilizam pesos sinápticos [12]. Nessa nomenclatura, as RAMs são consideradas como neurônios, e os valores dos contadores visitados pelas tuplas são utilizados para definir a ativação dos mesmos.

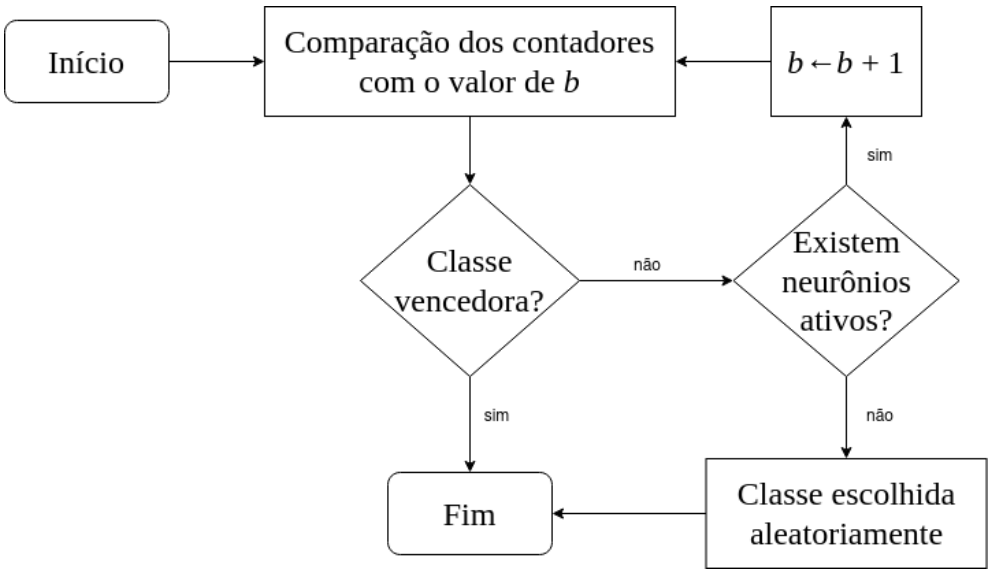


Figura 2.3: Fluxograma do *bleaching*.

O reconhecimento é feito através da apresentação da entrada para todos os discriminadores, com os valores lidos das memórias sendo passados para uma unidade

de decisão. Tal unidade realiza o processo descrito no fluxograma na Figura 2.3. O procedimento parte da comparação dos valores dos contadores com uma variável b, que inicialmente tem o valor igual a 1. Quando o valor do contador é maior ou igual a b o neurônio é considerado como ativo, e inativo caso contrário. A classe que obtém o maior número de neurônios ativos é retornada como a reconhecida pela rede e, caso ocorra empate entre duas ou mais classes, o valor de b é incrementado em uma unidade, e a comparação da quantidade de neurônios ativos é repetida. Esse processo de decisão é conhecido como *bleaching* (alvejante em inglês), e foi primeiro descrito em [13].

O nível de confiança da classificação pode ser calculado através da Equação 2.2, onde R_{max} é a maior pontuação encontrada e R_{2max} a segunda melhor. No processo de *bleaching*, caso nenhum neurônio responda como ativo, uma das categorias é escolhida aleatoriamente, e a confiança será igual a zero. A confiança provê um indicio da degradação gerada através do ruído no treinamento de forma que quão maior a confiança, mais improvável é que tenha ocorrido uma classificação errada [2].

$$\kappa = \frac{R_{max} - R_{2max}}{R_{max}} \tag{2.2}$$

2.4 DRASiW: imagens mentais

Além da abordagem de treino e classificação há a possibilidade de um processo reverso na rede graças a natureza do treinamento realizado no modelo. Utilizando o mapeamento definido para a rede, os endereços das memórias podem ser decodificados para as respectivas posições da retina. Tal extensão que possibilita a inversão do processo de treinamento é chamada de DRASiW e foi relatada em [14].

Com a DRASiW podemos gerar uma imagem para cada discriminador da rede que, conforme apresentado em [15], nos dá uma representação visual do que foi aprendido durante o treinamento e é obtida através do somatório dos contadores correspondentes para cada posição da retina no caso do bit presente no endereço ser diferente de 0. A imagem gerada é, no caso de um mapeamento exclusivo, equivalente ao somatório pixel a pixel de todas as entradas apresentadas para o discriminador. As imagens geradas pela DRASiW são conhecidas como imagens mentais.

2.5 Estratégias de mapeamento

Tradicionalmente, a divisão dos *pixels* da retina em uma rede WiSARD é feita em tuplas de mesmo tamanho que são definidas de maneira pseudo-aleatória e sem

repetição de endereços. Entretanto, apesar de ser a técnica padrão utilizada no modelo, esta é apenas uma das abordagens possíveis.

A não-exclusividade no mapeamento da rede, com repetição de endereços, foi descrita em [8] para o classificador de ênuplas. Essa técnica é comumente utilizada no modelo WiSARD quando o tamanho da ênupla desejado não possibilita uma divisão sem resto com o tamanho da entrada.

O uso de um algoritmo genético na definição do mapeamento de uma rede de ênuplas foi primeiro proposto por [16] que, utilizando apenas o procedimento de mutação nos experimentos, obteve uma melhora de até 10% na acurácia quando comparado com o mapeamento aleatório no *dataset* proprietário utilizado. Em [4] uma implementação para a WiSARD foi definida introduzindo um procedimento de *crossover* entre mapeamentos.

Em [15] é apresentada uma aplicação que utiliza diferentes tamanhos de tuplas em uma mesma rede. Tal medida foi tomada buscando um equilíbrio entre a maior generalização das tuplas pequenas com a especificidade de tuplas grandes. A definição do tamanho da tupla em que um endereço deveria estar era probabilístico de acordo com a distância do *pixel* com o centro da imagem.

O trabalho desenvolvido em [17] introduz o uso de um algoritmo de busca estocástica na definição de tuplas para o mapeamento de modelos baseados em ênuplas. A abordagem era possível graças ao sistema de recompensa e punição elaborado para calcular o desempenho das tuplas de forma individual, que posteriormente foi utilizado na modelagem do problema da busca de tuplas com otimização por enxame de partículas [18].

Na Tabela 2.1 é apresentado de forma resumida a melhora de acurácia informada em cada um dos trabalhos descritos. Em alguns casos tal informação não foi disponibilizada e, por sua vez, não estão presentes na tabela.

Tabela 2.1: Melhora percentual de acurácia das abordagens.

Abordagem	**Conjunto de dados**	**Melhora na acurácia (%)**
Algoritmo genético [16]	*Dataset* proprietário	Entre 7% e 10%
Algoritmo genético [4]	UCI *datasets* [19]	Entre 1,74% e 54,26%
Busca estocástica [17]	NIST [20]	3,38%
Otimização por enxame de partículas [18]	NIST [20]	5,02%

Capítulo 3

Meta-heurísticas de busca

No capítulo anterior foi apresentado o modelo WiSARD e os trabalhos relacionados à geração de mapeamentos. No presente capítulo, são evidenciados alguns conceitos necessários que são utilizados nos modelos propostos neste trabalho.

3.1 Busca estocástica de hiper-parâmetros para classificador de ênuplas

O trabalho apresentado por Azhar em [17] introduz uma abordagem de busca por um mapeamento que aumente a taxa de reconhecimento em classificadores baseados em ênuplas quando comparado com os gerados aleatoriamente. O procedimento é descrito no Algoritmo 1, e recebe como parâmetro uma lista **m** com a quantidade de RAMs específicas para cada classe no mapeamento que será gerado.

Algoritmo 1 Busca estocástica (Azhar)

Entrada: m

$i \leftarrow 1, t \leftarrow 1, \mathbf{M} \leftarrow \emptyset, \mathbf{G} \leftarrow \emptyset, \mathbf{P} \leftarrow \emptyset$
while $i \leq \text{LEN}(\mathbf{c})$ **do**
  $\mathbf{G}(t) \leftarrow \text{RANDOMTUPLES}(m_i - \text{LEN}(\mathbf{P}(t-1)))$ ▷ Gera tuplas aleatórias
  $\mathbf{G}(t) \leftarrow \text{APPEND}(\mathbf{G}(t), \mathbf{P}(t-1))$
  $\mathbf{P}(t) \leftarrow \text{THRESHOLDFUNC}(\mathbf{G}(t))$ ▷ Seleciona as melhores tuplas
  $t \leftarrow t + 1$
  if $\text{LEN}(\mathbf{P}(t)) = m_i$ **then** ▷ Salvando as tuplas no mapeamento final
    $\mathbf{M} \leftarrow \text{APPEND}(\mathbf{M}, \mathbf{P}(t))$
    $i \leftarrow i + 1, t \leftarrow 1, \mathbf{P} \leftarrow \emptyset$
  end if
end while

A busca é por um mapeamento **M** formado pela união dos conjuntos de m_i ênuplas "especialistas"para cada classe i. As tuplas são geradas de maneira aleatória e avaliadas com base em uma função objetivo, chamada pelo autor de sistema de

recompensa e punição, que leva em conta os exemplos da classe i na avaliação das ênuplas, buscando assim as m_i ênuplas "especialistas". A Tabela 3.1 descreve as variáveis usadas no Algoritmo 1.

Tabela 3.1: Lista de símbolos da busca estocástica (Azhar)

Símbolo	**Descrição**
m	Vetor com os tamanhos das ênuplas
M	Mapeamento gerado
t	Iteração
i	Índice da classe
$\mathbf{G}(t)$	Matriz de ênuplas criadas na iteração t
$\mathbf{P}(t)$	Matriz de ênuplas consideradas válidas pela função limiar

Ao apresentar um conjunto de dados a uma tupla j, obtemos três taxas distintas: (1) s_c, indicando a proporção dos padrões reconhecidos; (2) s_r, referente aos padrões rejeitados; (3) s_m, para os padrões classificados erroneamente. Pesos são definidos *a priori* indicando o grau de importância de cada evento e a pontuação de cada ênupla é calculada conforme mostra a Equação 3.1.

$$o_j = w_c s_c + w_r s_r + w_m s_m \tag{3.1}$$

O algoritmo de busca seleciona as tuplas que pontuam positivamente em relação a um limiar. Tal seleção é realizada através da regra descrita na Equação 3.2, onde $\max_k o_k$ é a pontuação máxima alcançada no conjunto de tuplas da atual iteração, δ é a taxa de aprendizado e t é a atual iteração da busca.

$$o_j \geq (\max_k o_k)(1 - \exp(-\delta/t)) \tag{3.2}$$

3.2 Otimização por enxame de partículas

A otimização por enxame de partículas, ou PSO (do inglês *Particle Swarm Optimization*), é uma técnica de otimização inspirada no comportamento coletivo dos animais, onde uma população de soluções candidatas é definida como um enxame de partículas que se movem pelo espaço de parâmetros [21]. Cada partícula possui sua própria posição e velocidade, com seu movimento sendo definido a cada iteração do algoritmo através de uma função objetivo e pela relação com o restante do enxame.

Em [22], Azhar propõe o uso de PSO no mapeamento de redes baseadas em ênuplas, dando continuidade ao trabalho desenvolvido na busca estocástica. Cada tupla é considerada uma partícula, e os endereços que a constituem formam as coordenadas de um ponto em $\mathbb{R}^n$. O sistema de recompensa e punição apresentado

na Equação 3.1 é utilizado como função objetivo e a Equação 3.2 define quais ênuplas são válidas para serem salvas no mapeamento final.

A busca inicia com a definição de posições e velocidades das partículas de forma aleatória. A cada iteração a velocidade de cada dimensão d de uma partícula j em $\mathbf{V}_{j,d}$ é atualizada conforme a Equação 3.3a, onde o vetor $\mathbf{g}$ é a melhor posição encontrada em todo o enxame, e a matriz $\mathbf{L}$ contém as melhores posições individuais de cada partícula. Além disso, as constantes ψ_l e ψ_g determinam a influência da inteligência individual e coletiva, respectivamente. Outro fator utilizado são as variáveis r_l e r_g, que introduzem um comportamento pseudo-aleatório, impedindo que uma partícula fique repetindo um mesmo caminho múltiplas vezes. O parâmetro ω é chamado de fator de inércia, determinando o peso da posição anterior da partícula no cálculo da nova localização.

$$\mathbf{V}_{j,d}(t+1) = \omega \mathbf{V}_{j,d}(t) + \psi_l r_l(\mathbf{L}_{j,d} - \mathbf{G}_{j,d}(t)) + \psi_g r_g(\mathbf{g}_d - \mathbf{G}_{j,d}(t)) \tag{3.3a}$$

$$\mathbf{G}_{j,d}(t+1) = \mathbf{G}_{j,d}(t) + \mathbf{V}_{j,d}(t+1) \tag{3.3b}$$

Após o valor da velocidade ser atualizado, a nova posição da partícula é calculada conforme apresentado na Equação 3.3b. Como as posições são os endereços presentes nas tuplas, o movimento das partículas é restrito ao intervalo dos endereços da retina. A Tabela 3.2 apresenta a descrição de cada símbolo usado na abordagem.

Tabela 3.2: Lista de símbolos da otimização por enxame de partículas (Azhar)

Símbolo	**Descrição**
$\mathbf{m}$	Vetor com os tamanhos das ênuplas
$\mathbf{M}$	Mapeamento gerado
n	Tamanho da tupla
μ	Tamanho inicial da população
t	Iteração
i	Índice da classe
$\mathbf{G}(t)$	Posição das partículas na iteração t
$\mathbf{V}(t)$	Velocidade das partículas na iteração t
$\mathbf{P}(t)$	Matriz de ênuplas consideradas válidas pela função limiar

No Algoritmo 2 é explicitado que, caso m_i partículas satisfaçam a condição imposta pela função de limiarização, mostrada na Equação 3.2, serão salvas no mapeamento final $\mathbf{M}$, e são consideradas como especialistas da classe i.

Posteriormente, no trabalho desenvolvido em [18], Azhar apresenta uma extensão do modelo que utiliza um sistema de criticidade auto-organizada na etapa da atualização das posições, que impede uma convergência prematura do PSO ao forçar uma diversidade mínima na população de soluções.

Algoritmo 2 Otimização por enxame de partículas (Azhar)

Entrada: $\mathbf{m}, n, \mu$

$i \leftarrow 1, t \leftarrow 1, \mathbf{M} \leftarrow \emptyset$
$\mathbf{G}(t) \leftarrow$ randomTuples(μ) ▷ Gera tuplas aleatórias
$\mathbf{V}(t) \leftarrow$ randomVelocities(μ) ▷ Gera velocidades aleatórias
while $i \leq$ len$(\mathbf{c})$ **do**
    $\mathbf{V}(t+1) \leftarrow$ updateVelocities$(\mathbf{V}(t), \mathbf{G}(t))$ ▷ Atualiza as velocidades
    $\mathbf{G}(t+1) \leftarrow$ updatePositions$(\mathbf{V}(t+1), \mathbf{G}(t))$ ▷ Atualiza as posições
    $\mathbf{P}(t+1) \leftarrow$ thresholdFunc$(\mathbf{G}(t+1))$ ▷ Seleciona as melhores tuplas
    $t \leftarrow t+1$
    if len$(\mathbf{P}(t)) \geq m_i$ **then** ▷ Salvando as tuplas no mapeamento final
        $\mathbf{M} \leftarrow$ append$(\mathbf{M}, \mathbf{P}(t))$
        $i \leftarrow i+1, t \leftarrow 1$
    end if
end while

Conforme representado na Figura 3.1, a posição de uma partícula é avaliada a cada iteração e, caso a posição de qualquer dimensão alcance uma certa criticidade acima de um limiar estabelecido, sua coordenada será alterada para um endereço vizinho na retina. O limiar informa a quantidade que um endereço pode ser repetido por todo o enxame e, desta forma, quão menor o valor escolhido, mais dispersões serão necessárias para alcançar o grau de diversidade desejado. O valor mínimo para que tal procedimento possa ser executado corretamente é igual a $\frac{n\mu}{\rho}$, onde ρ é igual ao número de *pixels* na retina.

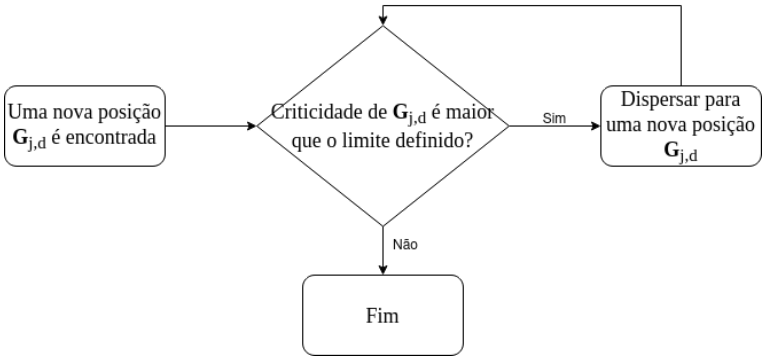


Figura 3.1: Fluxograma do sistema de criticidade auto-organizada.

3.3 Algoritmos Genéticos

Inspirado no processo de seleção natural, um algoritmo genético é uma meta-heurística que reflete a seleção dos indivíduos mais saudáveis para a recombinação, além de eventuais mutações [23]. Um indivíduo, também chamado de cromossomo, nada mais é que uma das possíveis soluções para o problema em questão, que é com-

posto por variáveis denominadas de genes. Um conjunto de indivíduos é chamado de população. Uma implementação de um algoritmo genético geralmente inicia com uma população aleatória com o intuito de gerar uma diversidade de soluções [24].

O processo de seleção se dá através de uma função objetivo, que define o nível de saúde de um indivíduo representando a probabilidade de um indivíduo ser escolhido para a recombinação. A partir disso, dois ou mais indivíduos são escolhidos para a etapa de cruzamento (do inglês *crossover*), também chamada de recombinação, que é o principal processo no qual novos cromossomos são criados. A principal ideia desta abordagem é que soluções descendentes herdarão as características dos indivíduos mais saudáveis, preservando e combinando suas características.

Certos indivíduos criados em uma geração podem ser submetidos a um processo de mutação visando manter uma diversidade genética, evitando uma convergência prematura. Mutações alteram um ou mais genes de um cromossomo, dada uma probabilidade definida *a priori*. Um exemplo deste procedimento é o *bit string mutation* que, dado um cromossomo formado por uma sequência binária, altera determinados *bits* de 0 para 1, ou o inverso, escolhidos aleatoriamente.

A solução apresentada por Giordano em [4] descreve um algoritmo genético na criação de mapeamentos para o modelo WiSARD. Para tal, utiliza a acurácia de uma rede treinada com o mapeamento informado como o nível de saúde utilizado no algoritmo genético. Sua estrutura é descrita no Algoritmo 3 e a Tabela 3.3 apresenta a descrição de cada um dos símbolos usados pelo algoritmo.

Tabela 3.3: Lista de símbolos do algoritmo genético (Giordano)

Símbolo	**Descrição**
$\mathbf{M}$	Mapeamento gerado
n	Tamanho da tupla
μ	Tamanho inicial da população
t	Iteração
$\mathbf{G}$	Descendência
$\mathbf{F}(t)$	Valores da saúde dos mapeamentos na iteração t
$\mathbf{P}(t)$	Matriz dos mapeamentos que compõem a população na iteração t
τ	Limitador de iterações t
λ	Tamanho da descendência t
θ_r	Limiar de recombinação t
θ_u	Limiar de mutação t

A inicialização do algoritmo se dá na criação da população inicial $\boldsymbol{P}$ com μ mapeamentos de tuplas com tamanho n. Um ponto a ser destacado na abordagem proposta é a utilização de mapeamentos sequenciais, ao invés de mapeamentos aleatórios, o que leva a uma falta de diversidade nas primeiras iterações, dependendo totalmente da diversidade criada durante o processo de mutação.

Algoritmo 3 Algoritmo genético (Giordano)

Entrada: $n, \mu, \tau, \lambda, \theta_r, \theta_u$

1: $t \leftarrow 1$
2: $\boldsymbol{P}(t) \leftarrow \text{INITIALIZE}(\mu, n)$ ▷ Cria a população inicial
3: $\boldsymbol{F}(t) \leftarrow \text{EVALUATE}(\boldsymbol{P}(t))$ ▷ Calcula a acurácia dos mapeamentos
4: **while** $t < \tau$ **do** ▷ Loop de evolução
5: $\quad\boldsymbol{G} \leftarrow \emptyset$
6: $\quad$**for** $i \leftarrow 1; i < \lambda; i \leftarrow i + 1$ **do**
7: $\quad\quad r \leftarrow \text{RANDOM}()$
8: $\quad\quad$**if** $r < \theta_r$ **then** ▷ Recombinação
9: $\quad\quad\quad\boldsymbol{M} \leftarrow \text{CROSSOVER}(\text{RANDOMPAIRINDEX}(\boldsymbol{P}(t)))$
10: $\quad\quad$**else**
11: $\quad\quad\quad$**if** $r < \theta_r + \theta_u$ **then** ▷ Mutação
12: $\quad\quad\quad\quad\boldsymbol{M} \leftarrow \text{MUTATION}(\text{RANDOMINDEX}(\boldsymbol{P}(t)))$
13: $\quad\quad\quad$**else** ▷ Cópia
14: $\quad\quad\quad\quad\boldsymbol{M} \leftarrow \text{CLONE}(\text{RANDOMINDEX}(\boldsymbol{P}(t)))$
15: $\quad\quad\quad$**end if**
16: $\quad\quad$**end if**
17: $\quad\quad\boldsymbol{G} \leftarrow \text{APPEND}(\boldsymbol{M}, \boldsymbol{G})$
18: $\quad$**end for**
19: $\quad t \leftarrow t + 1$
20: $\quad\boldsymbol{P}(t) \leftarrow \boldsymbol{G} + \boldsymbol{P}(t-1)$
21: $\quad\boldsymbol{F}(t) \leftarrow \text{EVALUATE}(\boldsymbol{P}(t))$
22: $\quad\boldsymbol{P}(t) \leftarrow \text{SELECT}(\boldsymbol{P}(t), \boldsymbol{F}(t), \mu)$ ▷ Seleciona μ indivíduos da população
23: **end while**

A seleção é feita durante as τ iterações do algoritmo, sendo em cada iteração criada uma descendência $\boldsymbol{G}$ de tamanho λ. O procedimento é realizado elegendo os μ mapeamentos mais saudáveis do conjunto formado pela descendência e da população da iteração passada.

Um novo mapeamento M é adicionado a descendência através da recombinação, mutação ou cópia de elementos selecionados aleatoriamente da população. A decisão sobre qual operação será executada é feita através do range definido pelas variáveis θ_r e θ_u, onde $0 \geq \theta_r + \theta_u \leq 1$. Um valor aleatório no intervalo $[0, 1]$ define qual ação será escolhida, de tal forma que: caso o valor seja menor que θ_r, a recombinação ocorre; o valor esteja no intervalo $[\theta_r, \theta_r + \theta_u)$, o processo de mutação ocorrerá; e, por último, caso o valor esteja no intervalo $[\theta_r + \theta_u, 1]$, um cromossomo é escolhido aleatoriamente e copiado para a descendência.

No processo de recombinação, dois mapeamentos são escolhidos aleatoriamente da população e são combinados através de um ponto de cruzamento (do inglês *crossover point*), conforme demonstrado na Figura 3.2, onde metade das tuplas de cada mapeamento são combinadas formando um terceiro. Caso o mapeamento gerado seja não-exclusivo, os endereços repetidos são substituídos.

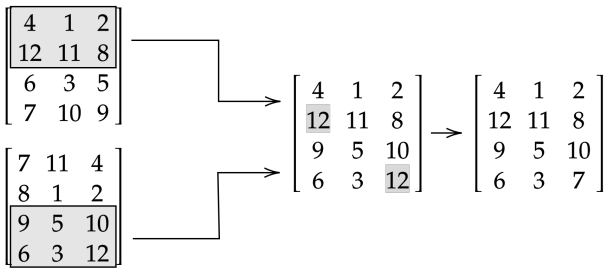


Figura 3.2: Processo de recombinação com mapeamentos (Giordano).

Na mutação dos cromossomos, no máximo n pares de ênuplas são selecionados de forma aleatória e, para cada par escolhido, há uma troca dos endereços na coluna indicada. Uma ilustração desse procedimento pode ser visto na Figura 3.3.

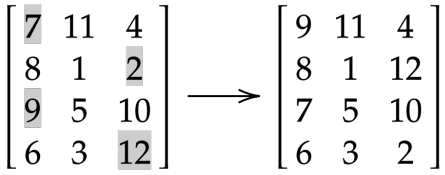


Figura 3.3: Processo de mutação em um mapeamento (Giordano).

3.4 Conceitos acessórios

3.4.1 Problema da satisfação de restrições ponderada

Em geral, um problema de satisfação de restrições (do inglês *Constraint Satisfaction Problem*) é composto por um conjunto de variáveis, onde cada variável possui um domínio finito de valores possíveis e um conjunto de restrições. Algoritmos de resolução de tais problemas buscam atribuir um valor a cada variável satisfazendo as restrições impostas [25].

Entretanto, existem problemas que não possuem soluções sem a quebra de uma ou mais restrições, ou cenários de busca da melhor solução entre múltiplas possíveis. Este tipo de problema, que é necessário a exposição de preferências entre as restrições é conhecido como problema de satisfação de restrições ponderada (do inglês *Weighted Constraint Satisfaction Problem*, WCSP) [26]. Neste cenário, cada restrição possui um peso associado, e a busca de uma solução se torna maximização ou minimização do somatório dos pesos [27].

A escolha do algoritmo de resolução de WCSP depende do domínio em que o problema está inserido e da modelagem adotada [28]. Entretanto, os métodos de *branch-and-bound* [29] e *bucket elimination* [30] são as abordagens mais populares.

3.4.2 Entropia

Introduzido por Shannon em 1948 [31], a entropia da informação, por vezes chamada apenas de entropia, é a quantidade de informação contida em uma variável aleatória. Dada uma variável aleatória X, a probabilidade de cada possível evento x_i é representada por $\mathrm{P}(x_i)$, e calculamos a entropia $\mathrm{H}(X)$ conforme demonstrado na Equação 3.4.

$$\mathrm{H}(X) = -\sum_{i} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i) \tag{3.4}$$

A entropia também é interpretada como a taxa média na qual as informações são produzidas por uma fonte estocástica de dados. Utilizando o lançamento de uma moeda como exemplo, no caso de uma moeda imparcial, a função de entropia atinge seu valor máximo e tende ao mínimo da função quão mais parcial for a moeda. A extensão de tais valores depende da medida da informação adotada, definida na base b, que é tipicamente medida em bits, correspondendo a base 2.

No caso de só existirem dois valores esperados mutualmente exclusivos, a probabilidade de um evento é igual a p e do outro igual a $1 - p$. Desta forma, temos a função de entropia binária (ver Equação 3.5), que é um caso especial da função de entropia definida na Equação 3.4.

$$\mathrm{H}(X) = \mathrm{H}_2(p) = -p \log_2 p - (1-p) \log_2 (1-p) \tag{3.5}$$

Capítulo 4

Metodologia

Neste capítulo, três diferentes abordagens para a criação dos mapeamentos são apresentadas e, utilizando os conceitos introduzidos no capítulo anterior, os algoritmos que as compõem são descritos. Em tais algoritmos encontramos adaptações de técnicas já utilizadas na literatura e que demonstram grande potencial, além de procedimentos inéditos para a criação de mapeamentos no modelo WiSARD.

4.1 Otimização Estocástica

Otimização estocástica diz respeito a um conjunto de métodos de otimização que utilizam variáveis aleatórias e uma função objetivo, que buscamos maximizar ou minimizar.

4.1.1 Busca Estocástica e otimização por enxame de partículas

Os algoritmos de busca estocástica e otimização por enxame de partículas desenvolvidos por Azhar, e apresentados no capítulo anterior, foram criados baseados no funcionamento do classificador de ênuplas. Portanto, algumas adaptações no sistema de recompensa e punição proposto são necessárias para que seja viável a utilização de tais modelos em uma WiSARD com *bleaching*. Tal modificação é necessária uma vez que as abordagens propostas focam em obter tuplas especialistas e independentes, desconsiderando o contexto e as comparações realizadas no algoritmo do *bleaching* em um mapeamento completo.

Na nova proposta cada mapeamento de ênupla, que está sendo avaliado, é aplicado em um conjunto de RAMs, em que cada classe é responsável por uma RAM, e os respectivos contadores são comparados. Ao apresentarmos um exemplo para o conjunto de RAMs, caso o maior valor encontrado seja da RAM da classe correta, consideramos que o padrão foi reconhecido. Quando a RAM com maior valor

de resposta não corresponde com a classe esperada o padrão é considerado como classificado erroneamente ou, no caso de empate, como rejeitado. Esse processo é repetido para cada exemplo do conjunto de treinamento e, desta forma, obtermos as taxas que são utilizadas para calcular a pontuação da tupla, conforme descrito na Equação 3.1.

Um algoritmo aprimorado de busca estocástica pode ser alcançado ao removermos a necessidade de especialização das ênuplas em classes específicas e, uma vez que m tuplas satisfaçam a condição imposta na Equação 3.2, temos um mapeamento $\mathbf{M}$ de uma rede WiSARD. Tal abordagem é descrita no Algoritmo 4.

Algoritmo 4 Algoritmo de busca estocástica (Aprimorado)

Entrada: τ, γ, n, m
1: $t \leftarrow 1$, $l \leftarrow 0$, $g \leftarrow 0$, $\mathbf{M} \leftarrow \emptyset$
2: **while** $t \leq \tau$ & $l < \gamma$ & LEN($\mathbf{M}$) $< m$ **do**
3: $\quad j \leftarrow 1$
4: $\quad$ **while** $j \leq m -$ LEN($\mathbf{M}$) **do**
5: $\quad\quad$ $\mathbf{M}$.INSERT(RANDOMTUPLE(n)) ▷ Gera tuplas aleatórias
6: $\quad\quad j \leftarrow j + 1$
7: $\quad$ **end while**
8: $\quad \mathbf{M} \leftarrow$ OBJECTIVEFUNC($\mathbf{M}$) ▷ Seleciona as melhores tuplas
9: $\quad t \leftarrow t + 1$
10: $\quad$ **if** LEN($\mathbf{M}$) $== g$ **then**
11: $\quad\quad l \leftarrow l + 1$
12: $\quad$ **else**
13: $\quad\quad g \leftarrow$ LEN($\mathbf{M}$)
14: $\quad\quad l \leftarrow 0$
15: $\quad$ **end if**
16: **end while**

Variáveis de controle das iterações foram adicionadas, onde τ determina a quantidade máxima de iterações e γ determina o número máximo de iterações em que é aceitável que não haja mudança no tamanho de ênuplas consideradas aceitáveis pelo algoritmo. Os símbolos usados no algoritmo aprimorado são apresentados na Tabela 4.1.

Tabela 4.1: Lista de símbolos da busca estocástica (Aprimorado)

Símbolo	Descrição
$\mathbf{M}$	Mapeamento gerado
n	Tamanho da tupla
m	Quantidade de ênuplas
t	Iteração
τ	Limitador de iterações
γ	Limitador de iterações sem alteração no valor médio da função objetivo

Outro ponto importante, adicionado a abordagem aprimorada, é uma nova taxa

s_f no cálculo da pontuação para o caso de um empate entre a classe correta e uma ou mais outras classes, sendo chamado de estado de insuficiência, em contraste ao evento de um empate em que não envolva a classe esperada.

4.1.2 Algoritmo Genético

A modelagem proposta por Giordano, relatada no Capítulo 3, utiliza um algoritmo genético na criação do mapeamento para redes WiSARD. Entretanto, alguns problemas foram identificados, como a população inicial sem diversidade e os valores da função objetivo não serem utilizados durante a seleção dos indivíduos. Para sanar tais problemas, uma abordagem aprimorada foi elaborada e é descrita no Algoritmo 5.

Algoritmo 5 Algoritmo genético (Aprimorado)

Entrada: $n, \mu, \tau, \gamma, \lambda, \theta$

1: $t \leftarrow 1, l \leftarrow 0$
2: $\boldsymbol{P}(t) \leftarrow \text{INITIALIZE}(\mu, n)$ ▷ Cria a população inicial
3: $\boldsymbol{F}(t) \leftarrow \text{EVALUATE}(\boldsymbol{P}(t))$ ▷ Calcula a acurácia dos mapeamentos
4: **while** $t < \tau$ & $l < \gamma$ **do** ▷ Loop de evolução
5: $\boldsymbol{G} \leftarrow \emptyset$
6: **for** $i \leftarrow 1; i < \lambda; i \leftarrow i + 1$ **do**
7: $r \leftarrow \text{RANDOM}()$
8: **if** $r < \theta$ **then** ▷ Recombinação
9: $\boldsymbol{M_1}, \boldsymbol{M_2} \leftarrow \text{CROSSOVER}(\text{RANDOMPAIRINDEX}(\boldsymbol{P}(t)))$
10: $\boldsymbol{G} \leftarrow \text{APPEND}(\boldsymbol{M_2}, \boldsymbol{G})$
11: **else** ▷ Mutação
12: $\boldsymbol{M_1} \leftarrow \text{MUTATION}(\text{RANDOMINDEX}(\boldsymbol{P}(t)))$
13: **end if**
14: $\boldsymbol{G} \leftarrow \text{APPEND}(\boldsymbol{M_1}, \boldsymbol{G})$
15: **end for**
16: $t \leftarrow t + 1$
17: $\boldsymbol{P}(t) \leftarrow \boldsymbol{G} + \boldsymbol{P}(t-1)$
18: $\boldsymbol{F}(t) \leftarrow \text{EVALUATE}(\boldsymbol{P}(t))$
19: $\boldsymbol{P}(t), \boldsymbol{F}(t) \leftarrow \text{SELECT}(\boldsymbol{P}(t), \boldsymbol{F}(t), \mu)$ ▷ Seleciona μ indivíduos da população
20: **if** $\text{MEAN}(\boldsymbol{F}(t)) = \text{MEAN}(\boldsymbol{F}(t-1))$ **then**
21: $l \leftarrow l + 1$
22: **else**
23: $l \leftarrow 0$
24: **end if**
25: **end while**

Uma das principais modificações propostas é a simplificação das condições para a recombinação e mutação dos indivíduos, onde a variável θ funciona como o limiar de decisão dado um valor aleatório r. A decisão de remover o procedimento de cópia de um elemento foi tomada uma vez que não acrescenta nenhum elemento inédito

na população da nova geração.

Semelhantemente ao que foi apresentado no algoritmo aprimorado de busca estocástica, τ indica o número máximo de iterações do algoritmo, enquanto γ restringe o número de iterações em que o algoritmo pode continuar rodando sem uma alteração significativa na população, verificação feita através da comparação da média das acurácias dos mapeamentos. A lista de símbolos usados no algoritmo genético aprimorado é apresentada na Tabela 4.2.

Tabela 4.2: Lista de símbolos do algoritmo genético (Aprimorado)

Símbolo	**Descrição**
$\mathbf{M}$	Mapeamento gerado
n	Tamanho da tupla
μ	Tamanho inicial da população
t	Iteração
$\mathbf{G}$	Descendência
$\mathbf{F}(t)$	Valores da saúde dos mapeamentos na iteração t
$\mathbf{P}(t)$	Matriz dos mapeamentos que compõem a população na iteração t
τ	Limitador de iterações
γ	Limitador de iterações sem alteração no valor médio da função objetivo
θ	Limiar de recombinação
λ	Tamanho da descendência

O procedimento de recombinação foi alterado, removendo a verificação de endereços duplicados, permitindo assim a criação de mapeamentos não-exclusivos. Outra modificação foi realizada na quantidade de mapeamentos gerados no processo que, diferentemente da abordagem apresentada por Giordano, permite que dois mapeamentos $\mathbf{M_1}$ e $\mathbf{M_2}$ sejam gerados a partir da combinação das metades dois mapeamentos originais, conforme ilustrado na Figura 4.1.

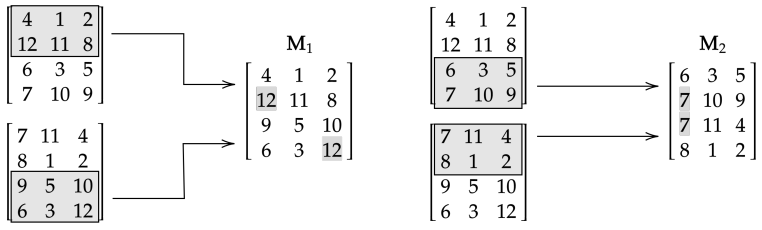


Figura 4.1: Processo de recombinação com mapeamentos (Aprimorado).

A mutação dos mapeamentos também foi modificada onde, com uma probabilidade $\frac{1}{m}$, todos os endereços em uma ênupla são modificados para novos valores escolhidos aleatoriamente.

4.2 Estratégias baseadas nas imagens mentais

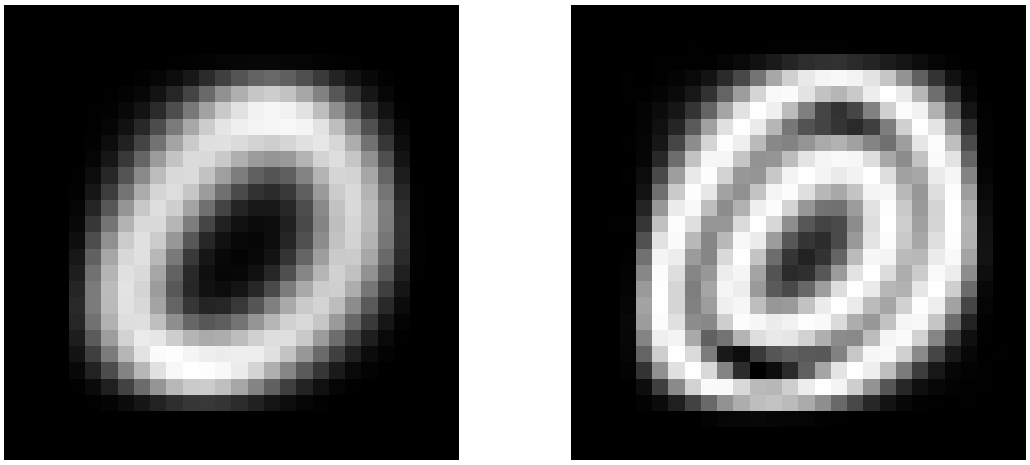
Conforme apresentado no Capítulo 2, a imagem mental funciona como uma representação visual do que foi aprendido por um discriminador durante o treinamento e, além disso, apresenta de forma concisa o conhecimento presente sobre uma determinada classe no conjunto de treinamento.

Se partirmos do pressuposto que os *pixels* da retina de um discriminador são variáveis aleatórias independentes, podemos considerar a intensidade normalizada da imagem mental como a probabilidade de um determinado endereço ter valor igual a 1 em um padrão. A Equação 4.1 apresenta a normalização adotada, onde v_i é a i-ésima posição da imagem mental e p_i é a probabilidade calculada.

$$p_i = \frac{v_i}{\max_i v_i} \tag{4.1}$$

Conforme apresentado no Capítulo 3, podemos estabelecer o nível médio de incerteza atrelado a um possível resultado da variável aleatória por meio do cálculo da entropia e, para o caso especial de uma variável de Bernoulli, através da função de entropia binária, apresentada na Equação 3.5.

Na Figura 4.2 é possível notar a diferença de uma imagem mental para uma imagem de entropia geradas a partir da classe "0"do conjunto de dados MNIST com limiarização pelo valor-médio. A imagem de entropia é uma representação visual da entropia calculada a partir das probabilidades obtidas na imagem mental.



(a) Imagem mental (b) Imagem de entropia

Figura 4.2: Comparação entre uma imagem mental e uma imagem da entropia.

4.2.1 Mapeamento por agrupamento

Podemos utilizar o conhecimento das entropias calculadas a partir das imagens mentais para criar mapeamentos. Para realizar tal tarefa, o método de criação de mape-

amentos independentes parte do agrupamento dos endereços pelos seus respectivos valores. A Figura 4.3 ilustra o processo de criação de um mapeamento com $n = 3$, com o agrupamento através da ordenação dos endereços pelas respectivas entropias e, caso ocorra de dois ou mais endereços possuírem o mesmo valor, a prioridade de desempate é definida aleatoriamente.

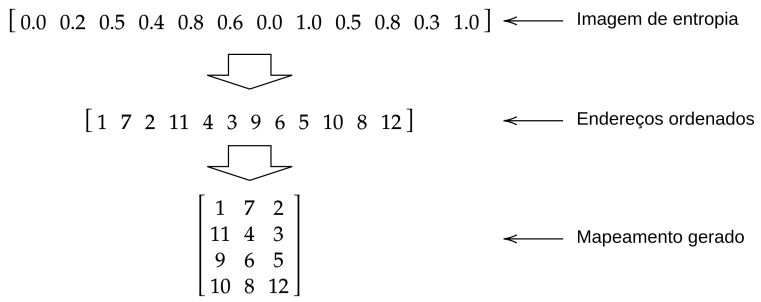


Figura 4.3: Criação de um mapeamento pelo agrupamento da imagem de entropia.

A ideia por trás dessa abordagem é criar um mapeamento que o número de endereços acessados é proporcional a entropia dos *pixels* que compõem a ênupla. Desta forma, quão menor for a entropia dos endereços, menos posições na RAM terão seus contadores com valor diferente de zero. O caso contrário, quando a entropia dos endereços é alta, apesar de ser provável, não necessariamente resulta em mais posições acessadas durante o treinamento.

Um mapeamento compartilhado pode ser criado utilizando o mesmo sistema de agrupamento. Entretanto, no lugar da imagem de entropia, utilizamos ferramentas de estatística descritiva para sumarizar as diferentes imagens de entropia em apenas uma que descreva em algum nível a relação entre as diferentes classes como, por exemplo, a média, desvio-padrão, curtose ou obliquidade pixel-a-pixel das diferentes imagens.

4.3 Mapeamento por restrições

A criação de mapeamentos através de processos de otimização estocástica ou com a entropia calculada utilizam o conjunto de treinamento em diferentes níveis de abstração. O primeiro método emprega o conjunto para validar os mapeamentos e calcular métricas que servem como função objetivo do problema. A segunda técnica já parte para uma outra camada, criando mapeamentos através da consolidação dos dados. Entretanto, nenhum método apresentado até o momento avalia as correlações e padrões que ocorrem a cada exemplo disponível para o treinamento da rede.

Pensando nisso, um procedimento foi desenvolvido com a criação dos mapeamentos sendo modelada como um problema da satisfação de restrições ponderada, tema apresentado no Capítulo 3. Os endereços da retina são as variáveis utilizadas,

e cada variável tem como domínio de possíveis valores as ênuplas a serem populadas durante a criação do mapeamento.

Uma técnica para a criação das restrições a partir do conjunto de treinamento foi elaborada, necessitando do conhecimento do domínio e das características do *dataset* para criar uma ou mais regras, que são descritas para o algoritmo através de padrões a serem buscados nos exemplos, com um peso associado a cada padrão definido. Toda vez que um dos padrões for encontrado em um exemplo, restrições são criadas com pares formados da combinação dos índices dos *pixels* envolvidos, onde cada par possui um valor correspondente ao somatório dos pesos.

O Algoritmo 6 apresenta a abordagem de extração das restrições, onde $\mathbf{X}$ é o conjunto de dados de uma determinada classe, e FINDER é o método que identifica os endereços ativos de acordo com as regras definidas. Os demais símbolos usados são apresentados na Tabela 4.3.

Algoritmo 6 Extração das restrições

Entrada: $\mathbf{X}$, FINDER

$\mathbf{R} \leftarrow \{\}$ ▷ Dicionário de restrições
for $i \leftarrow 1; i \leq \text{LEN}(\mathbf{X}); i \leftarrow i + 1$ **do**
  $\mathbf{E} \leftarrow \text{FINDER}(\mathbf{X}_i)$ ▷ Encontra os endereços ativos para cada regra
  for $j \leftarrow 1; j \leq \text{LEN}(\mathbf{E}); j \leftarrow j + 1$ **do**
    $\mathbf{P} \leftarrow \text{PAIRCOMBINATIONS}(\mathbf{E}_j)$ ▷ Combinação de pares de endereços
    for $k \leftarrow 1; k \leq \text{LEN}(\mathbf{P}); k \leftarrow k + 1$ **do**
      $\mathbf{P}_k \leftarrow \text{SORT}(\mathbf{P}_k)$
      $\mathbf{R}(\mathbf{P}_{k,1}, \mathbf{P}_{k,2}) \leftarrow \mathbf{R}(\mathbf{P}_{k,1}, \mathbf{P}_{k,2}) + 1$
    end for
  end for
end for

Tabela 4.3: Lista de símbolos do mapeamento por restrições

Símbolo	Descrição
$\mathbf{R}$	Dicionário de restrições
FINDER	Método que identifica os endereços ativos das regras definidas
$\mathbf{X}$	Conjunto de dados de uma determinada classe
$\mathbf{E}$	Endereços ativos
$\mathbf{P}$	Pares de endereços ativos
$\mathbf{e}$	Vetor com as prioridades
n	Tamanho da tupla
m	Quantidade de ênuplas
$\mathbf{M}$	Mapeamento gerado
$\mathbf{P}$	Restrições não satisfeitas

Como exemplo, definiremos uma regra de "separação dos uns", que buscará em cada item por dois ou mais *pixels* com valor igual a 1 e criará restrições indicando

que tais índices não devem estar nas mesmas ênuplas de um mapeamento. Tomemos o conjunto de exemplos $\mathbf{X} = \{0110, 1010, 0110, 1110\}$, com o peso do evento sendo igual a 1, a Tabela 4.4 apresenta as restrições encontradas e os somatórios dos pesos associados.

Tabela 4.4: Exemplo de restrições extraídas de um conjunto de dados.

Restrição	(2, 3)	(1, 3)	(1, 2)
Peso	3	2	1

Após a extração das restrições, uma ordem de escolha entre os índices que serão usados no mapeamento precisa ser definida. Para isso, a prioridade de escolha de um endereço é definida como o somatório dos pesos das restrições em que ele aparece. O Algoritmo 7 detalha o cálculo da prioridade de escolha, onde $\mathbf{R}$ é o dicionário de restrições.

Algoritmo 7 Definição das prioridades de escolha

Entrada: $\mathbf{R}$, n, m

1: $\mathbf{e} \leftarrow \emptyset$ ▷ Vetor com as prioridades
2: **for** $i \leftarrow 1; i \leq n \times m; i \leftarrow i + 1$ **do**
3: $\quad \mathbf{e}_i \leftarrow 0$ ▷ Zera a prioridade do endereço
4: **end for**
5: **for** k_1, k_2 **in R do** ▷ Para cada par nas restrições
6: $\quad \mathbf{e}_{k_1} \leftarrow \mathbf{e}_{k_1} + \mathbf{R}(k_1, k_2)$
7: $\quad \mathbf{e}_{k_2} \leftarrow \mathbf{e}_{k_2} + \mathbf{R}(k_1, k_2)$
8: **end for**

Seguindo o exemplo apresentado anteriormente do conjunto $\mathbf{X}$, temos a Tabela 4.5 que apresenta as prioridades de escolhas calculadas.

Tabela 4.5: Exemplo de prioridades de escolha geradas a partir das restrições.

Índice	1	2	3	4
Prioridade	3	4	5	0

A criação de um mapeamento se dá com a utilização das prioridades de escolha calculadas previamente na formação das tuplas. Os endereços são selecionados em ordem decrescente de prioridade e designados a uma tupla específica com base nos critérios definidos a seguir:

1. Seleciona a tupla disponível com o menor somatório dos pesos das restrições;
2. Caso haja empate no primeiro item, seleciona dentre as tuplas que geraram o empate a que possui a menor quantidade de restrições a serem quebradas;

3. Caso haja empate nos itens anteriores, seleciona aleatoriamente uma tupla entre as definidas no segundo item.

A disponibilidade de uma tupla é dada pela quantidade de endereços designados a ela ser menor que o tamanho de tupla definido. No Algoritmo 8 é apresentada a criação do mapeamento $\mathbf{M}$ a partir de um vetor de prioridades $\mathbf{e}$, um dicionário de restrições $\mathbf{R}$, e o tamanho e quantidade de ênuplas n e m, respectivamente.

Algoritmo 8 Criação do mapeamento

Entrada: $\mathbf{e}$, $\mathbf{R}$, n, m

1: $\mathbf{M} \leftarrow \emptyset$
2: $\mathbf{e} \leftarrow$ REVERSEARGSORT($\mathbf{e}$)
3: **for** $i \leftarrow 1; i \leq$ LEN(e); $i \leftarrow i + 1$ **do**
4: $\quad \mathbf{P} \leftarrow$ PENALTYFUNC($e_i, \mathbf{R}, \mathbf{M}$) ▷ Calcula as restrições não satisfeitas em cada tupla, dado o endereço e_i
5: $\quad l \leftarrow$ TUPLESELECTION($\mathbf{M}, \mathbf{P}$,n,m) ▷ Seleciona a melhor tupla com base nos critérios definidos
6: $\quad \mathbf{M}_l$.INSERT(e_i)
7: **end for**

A Figura 4.4 apresenta um fluxograma que resume as etapas da abordagem. É importante ressaltar que a extração de restrições e a definição das prioridades de escolha só precisam ocorrer uma vez para um conjunto de dados, independente dos tamanhos das tuplas dos mapeamentos que serão criados.

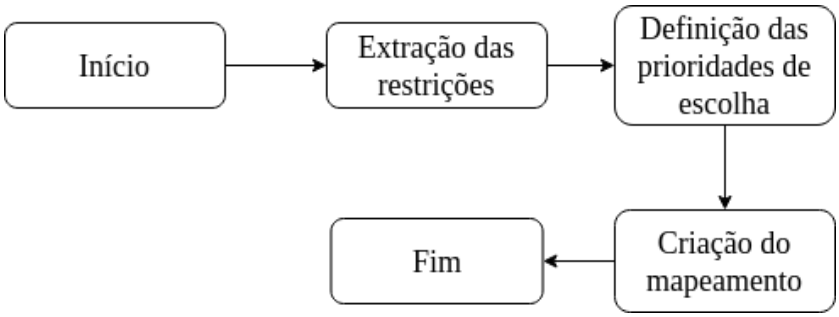


Figura 4.4: Fluxograma da criação de um mapeamento através das restrições.

Alguns procedimentos foram elaborados para a criação de mapeamentos compartilhados, utilizando três operações para determinar quais restrições extraídas dos exemplos das classes são utilizadas na definição das prioridades de escolha que, por sua vez, serão utilizados na criação de um mapeamento para toda a rede. Primeiro, temos a união dos conjuntos, que utiliza todas as restrições disponíveis e, caso uma restrição esteja presente em mais de um conjunto, seus pesos são somados. A segunda operação é a interseção, que procura quais restrições estão presentes em todas as classes, e utiliza a média dos pesos das restrições. Por último, temos a interseção estrita, que busca as restrições que aparecem em todos os conjuntos com o mesmo peso.

Capítulo 5

Experimentos

Neste capítulo serão relatados os resultados do modelo WiSARD com as abordagens definidas no capítulo anterior. Para tal, os *datasets* utilizados são descritos, juntamente com as binarização adotadas em cada um dos casos. As acurácias dos mapeamentos aleatórios são empregados como um padrão de comparação para avaliarmos a eficácia dos algoritmos propostos.

5.1 Conjuntos de dados

Os conjuntos de dados escolhidos para o presente trabalho podem ser divididos em dois grupos: *Datasets* de imagens e *Datasets* com dados esparsos. As binarizações definidas levam em conta tais diferenças.

5.1.1 *Datasets* de imagens

A seguir, são descritos os *datasets* formados por imagens que foram selecionados para a realização dos experimentos. A escolha de tais conjuntos se deu pela ampla utilização dos mesmos em testes e comparações de técnicas na área de aprendizado de máquina.

O conjunto de dados MNIST [32] é composto por imagens de dígitos manuscritos normalizados por tamanho e centralizados em uma imagem de tamanho fixo de 28x28 *pixels*. São 10 diferentes categorias, que compõem um conjunto de treinamento de 60000 exemplos e um conjunto de testes com 10000 exemplos.

Inspirado no MNIST, o Fashion-MNIST [33] é formado por imagens em escala de cinza centralizadas, possuindo a mesma quantidade de classes e exemplares. Entretanto, o conjunto é formado por imagens de peças de roupas e acessórios.

O conjunto de dados CIFAR-10 [34] contém 60000 imagens coloridas de 32x32 em 10 categorias diferentes: aviões, carros, pássaros, gatos, cervos, cães, sapos, cavalos,

navios e caminhões. São 50000 imagens para treino e 10000 para teste, onde cada classe é composta por 6000 imagens.

O conjunto de dados Extended Cohn-Kanade [35], também chamado de CK+, é contém 593 sequências de imagens capturadas de 123 indivíduos. Entretanto, somente 327 dessas sequências são utilizadas para classificação de emoções no *dataset*, onde apenas a imagem final da sequência representa a emoção em seu pico de expressão. No total, sete classes estão presentes, sendo 45 imagens de raiva, 18 de desprezo, 59 de desgosto, 25 de medo, 69 de felicidade, 28 de tristeza e 83 de surpresa. Uma vez que os dados não estão separados em treino e teste, uma divisão estratificada aleatória foi feita, resultando em um subconjunto com 261 imagens para treino e 66 imagens de teste.

Os métodos de binarização adotados em todos os conjuntos de dados descritos foram o de limiarização global pelo valor-médio da imagem, limiarização local, que calcula limiares locais com base na vizinhança dos *pixels*, e o método de Otsu [36], que determina um limiar global ótimo a partir do histograma da imagem. A Figura 5.1 apresenta o resultado das diferentes binarizações aplicadas em uma imagem do *dataset* CK+.

(a) Valor-médio (b) Local (c) Otsu

Figura 5.1: Comparação entre diferentes binarizações.

5.1.2 *Datasets* esparsos

Datasets tabulares também foram escolhidos para os experimentos, com os dados dos conjuntos sendo esparsos. Para o processo de binarização, o método de termômetro binário foi utilizado, em que uma sequência de *bits* é criada para cada *feature*, sendo uma escala baseada na distância de Hamming onde, quão maior for o valor da variável, mais *bits* na sequência terão valor igual a 1 [37].

O *dataset* IMDb [38] é formado por 50000 comentários de filmes retirados do site de mesmo nome. Cada comentário possui uma categoria referente ao sentimento do autor em relação ao filme, podendo ser positivo ou negativo. Os comentários foram pré-processados utilizando TF-IDF (do inglês *Term Frequency Inverse Document Frequency*), que calcula valores para cada palavra em um comentário por meio de

uma proporção inversa da frequência da palavra em um comentário específico para a porcentagem de comentários em que a palavra aparece [39]. Além disso, uma análise de variância foi feita para selecionar as palavras mais relevantes, reduzindo de mais de 10000 palavras para 1595. A divisão de treino e teste é feita de forma estratificada, com 25000 itens para cada subconjunto.

Formado por 100000 notas dadas para 1700 filmes por 943 diferentes usuários, sendo 273 mulheres e 670 homens, temos o conjunto de dados MovieLens. O objetivo é definir o sexo do usuário com base nas notas por ele dadas, que variam de 1 até 5 estrelas [40].

A esparsidade em um *dataset* apresenta a proporção dos elementos diferentes de zero pelo número total de elementos. Desta forma, após o pré-processamento e binarização, o conjunto de dados IMDb tem esparsidade de 97.6%, e 95.6% para o MovieLens.

5.2 Resultados

Os resultados aqui apresentados são as médias das acurácias de 20 treinamentos e classificações independentes para cada combinação de gerador de mapeamento, tamanho da tupla, *dataset* e binarização. Os tamanhos da tupla explorados foram os que não possuem resto na divisão com o tamanho das entradas de cada *dataset*.

As tabelas com os resultados dos experimentos apresentam as maiores médias para cada *dataset* nos diferentes tipos de mapeamentos gerados, com a melhor abordagem em cada categoria de experimento sendo destacada. As informações das binarizações utilizadas não são apresentadas e, de forma similar, o tamanho da tupla referente ao melhor resultado também é omitido. Essas informações, bem como o desvio padrão das acurácias dos experimentos, estão disponíveis no Apêndice A.

Os algoritmos foram implementados utilizando uma combinação de duas linguagens de programação. O modelo WiSARD e tarefas relacionadas a seu funcionamento interno, como o *Bleaching*, foram utilizados conforme consta na biblioteca wisardpkg [41], implementados na linguagem C++. Em contrapartida, a linguagem Python foi usada para a criação das demais partes, incluindo a modelagem das técnicas de criação de mapeamentos, devido a sua flexibilidade na modelagem a manuseio dos dados. Os modelos juntamente com os testes executados estão disponíveis em [42]. Os modelos apresentados da literatura também foram programados em Python, tendo em vista que nenhuma implementação foi disponibilizada nos trabalhos originais.

Os experimentos foram executados em paralelo em uma máquina com o sistema operacional Ubuntu 20.04 LTS, processador AMD Ryzen 5 3400g e 16GB de memória RAM. Os tempos de criação dos mapeamentos na especificação dada também são

apresentados de forma sumarizada no presente capítulo, e detalhadas no Apêndice A.

5.2.1 Mapeamentos Aleatórios

Na Tabela 5.1 são apresentados os resultados utilizando os mapeamentos aleatórios independentes e compartilhados. Tais valores serviram como um guia para validar as alternativas desenvolvidas. É importante destacar que, com exceção dos conjuntos de dados CK+ e MovieLens, o mapeamento aleatório compartilhado apresenta melhores resultados quando comparado com o independente. Tal comportamento pode estar relacionado as características em comum de ambos os conjuntos de dados, como o número reduzido de exemplares e o baixo número de classes, além do desbalanceamento na quantidade de itens por classe.

Tabela 5.1: Resultados utilizando mapeamentos aleatórios. MAC - Mapeamento Aleatório Compartilhado. MAI - Mapeamento Aleatório Independente.

Método **Dataset**	MAC	MAI
MNIST	93.84%	93.33%
Fashion	82.28%	80.71%
CK+	55.83%	56.29%
Cifar10	33.83%	32.20%
IMDb	64.41%	60.35%
MovieLens	68.92%	69.26%

De forma semelhante, na Tabela 5.2 são apresentados os tempos para a criação dos mapeamentos e instanciação dos mesmos em uma rede WiSARD. Apesar desses valores também serem usados como base de comparação com as abordagens propostas, é válido destacar que é improvável que qualquer outro método seja mais eficiente em tempo de criação, uma vez que não há processamento além da definição em ordem aleatória dos índices da retina no mapeamento das tuplas.

Tabela 5.2: Tempo de criação dos mapeamentos aleatórios (em milissegundos). MAC - Mapeamento Aleatório Compartilhado. MAI - Mapeamento Aleatório Independente.

Método **Dataset**	MAC	MAI
MNIST	0.08	1.26
Fashion	0.08	1.06
CK+	0.25	1.36
Cifar10	0.09	0.23
IMDb	0.24	1.02
MovieLens	0.19	1.07

Os resultados dos tempos de criação dos mapeamentos aleatórios apresentam novamente vantagem na utilização de mapeamentos compartilhados entre diferentes discriminadores. A razão de tal diferença é óbvia, uma vez que apenas um mapeamento precisará ser criado, independente do número de classes.

5.2.2 Otimização estocástica

Na criação de mapeamentos por métodos de otimização estocástica, os conjuntos de treinamento foram divididos em treinamento e validação, com 30% dos dados sendo utilizados nos procedimentos de validação nas funções objetivo de cada modelo. Desta forma, as acurácias foram obtidas com o mapeamento gerado por cada abordagem no conjunto de teste que, por sua vez, só é usado durante essa tarefa.

Os valores definidos para os parâmetros dos modelos foram fixados, de forma que, independente do conjunto de dados apresentado, os algoritmos foram executados com os mesmos valores. Nos modelos que utilizam o sistema de recompensa e punição, descrito no Capítulo 3, os valores $w_c = 39$, $w_r = -0.5$ e $w_m = -1$ foram definidos para os pesos das taxas dos padrões reconhecidos, rejeitados e classificados erroneamente, respectivamente. Tais valores foram escolhidos com base nos cálculos apresentados em [17] e, da mesma forma, a taxa de aprendizado δ foi fixada com valor igual a 100. Além disso, no modelo aprimorado de busca estocástica, o peso da taxa dos casos de empate envolvendo a classe correta w_f foi definido como 19.5, metade do valor usado para a taxa de padrões reconhecidos.

Os parâmetros do modelo de otimização por enxame de partículas (PSO) foram definidos de acordo com os valores apresentados em [18], com a inteligência individual como $\psi_l = 1$, a inteligência coletiva $\psi_g = 1$ e o fator de inércia $\omega = 0.2$. O número máximo de iterações da busca estocástica e do PSO foi estabelecido como 1000 e, no modelo aprimorado de busca estocástica, o *lag* de mudança das últimas iterações fixado como 10.

Nos algoritmos genéticos, o número máximo de iterações escolhido foi o de 100, o tamanho da população inicial como 10 e, conforme proposto em [4], o tamanho da descendência γ como o mesmo tamanho da população inicial. Os parâmetros θ_r e θ_u foram fixados em 0.5 e 0.2, respectivamente, e com $\theta = 0.8$, no caso do algoritmo genético aprimorado.

Dos modelos aprimorados propostos, a busca estocástica aprimorada se destaca ao alcançar resultados melhores não só quando comparada com a busca estocástica apresentada por Azhar, como também dos demais modelos de otimização estocástica e dos mapeamentos aleatórios, com exceção dos experimentos no *dataset* MovieLens. Em todos os casos das melhores abordagens, destacadas na Tabela 5.3, superaram as abordagens de criação de mapeamentos aleatórios.

Tabela 5.3: Resultados dos experimentos de otimização estocástica. PSO - *Particle Swarm Optimization* (Otimização por Enxame de Partículas). BE - Busca Estocástica (Azhar). BEA - Busca Estocástica (Aprimorado). AG - Algortimo Genético (Girodano). AGA - Algoritmo Genético (Aprimorado). MAC - Mapeamento Aleatório Compartilhado. MAI - Mapeamento Aleatório Independente.

Método **Dataset**	PSO	BE	BEA	AG	AGA	MAC	MAI
MNIST	92.37%	92.89%	95.08%	94.00%	94.17%	93.84%	93.33%
Fashion	78.05%	81.18%	82.84%	81.95%	82.29%	82.28%	80.71%
CK+	61.14%	57.35%	61.29%	53.86%	55.83%	55.83%	56.29%
Cifar10	30.73%	32.83%	33.91%	30.62%	33.59%	33.83%	32.20%
IMDb	62.21%	64.83%	68.59%	64.02%	64.34%	64.41%	60.35%
MovieLens	69.76%	69.52%	68.52%	71.61%	69.39%	68.92%	69.26%

Dentre os modelos com abordagens mais similares, dois fatores diferenciam o modelo aprimorado de busca estocástica. A especialização das ênuplas para cada classe proposta por Azhar na busca estocástica e no PSO parece não ser tão eficaz com uma WiSARD com *bleaching*, uma vez que abre mão do contexto de comparação entre as classes, que é tão importante no procedimento de classificação. Outra diferença está na utilização de uma quarta taxa definida no sistema de recompensa e punição, indicando um estado de empate envolvendo a classe correta que, apesar de não ser o cenário ideal, é mais desejável do que um não reconhecimento ou até mesmo uma classificação errada.

O algoritmo genético aprimorado proposto não alcançou nenhum resultado com a maior acurácia entre as abordagens de otimização estocástica. Entretanto, ao compararmos os resultados com os mapeamentos gerados de forma aleatória, podemos visualizar melhores valores alcançados, principalmente nos conjuntos de dados com um maior volume de exemplos. Quando contrastado com as abordagens propostas na literatura, o modelo se mostra competitivo, principalmente contra o algoritmo equivalente proposto por Giordano, que só alcança uma acurácia maior que o algoritmo aprimorado no conjunto de dados MovieLens. É importante destacar que com os algoritmos genéticos propostos obtemos uma população de mapeamentos candidatos. No caso dos experimentos aqui apresentados, somente o mapeamento da população final com o maior valor na função objetivo era utilizado.

Ser agnóstico as características dos conjuntos de dados apresentados é a principal característica dos modelos de otimização estocástica definidos no presente trabalho. Tal característica é relevante quando precisamos definir um mapeamento de uma WiSARD especificamente para um *dataset*, mas desconhecemos a natureza dos dados. Entretanto, há um custo inerente a esse aspecto que se faz presente no tempo de criação dos mapeamentos, apresentados na Tabela 5.4, onde há um aumento significativo em todos os modelos quando comparados com os mapeamentos aleatórios.

Tabela 5.4: Tempo de criação dos mapeamentos gerados pelos algoritmos de otimização estocástica (em milissegundos). PSO - *Particle Swarm Optimization* (Otimização por Enxame de Partículas). BE - Busca Estocástica (Azhar). BEA - Busca Estocástica (Aprimorado). AG - Algortimo Genético (Girodano). AGA - Algoritmo Genético (Aprimorado).

Método **Dataset**	PSO	BE	BEA	AG	AGA
MNIST	327307.58	42804.63	52213.80	205192.98	147724.87
Fashion	264524.26	46435.27	46590.22	569841.03	218820.77
CK+	148318.58	40099.80	24880.13	2522.83	15638.49
Cifar10	669417.50	145574.28	83671.84	1523520.83	492738.26
IMDb	556508.75	50353.75	127885.04	130532.24	211406.33
MovieLens	39125.70	39584.86	3823.64	2220.18	5829.13

Além de ser o modelo com as melhores acurácias, a busca estocástica aprimorada também está entre os métodos mais rápidos dessa categoria. Isso é possível devido a simplicidade do método aliado ao cálculo realizado no sistema de recompensa e punição ser menos custoso computacionalmente do que a classificação dos padrões, realizada na função de *fitness* dos algoritmos genéticos.

5.2.3 Mapeamento por agrupamento da entropia

A criação de mapeamentos através do agrupamento da entropia foi aplicada para gerar mapeamentos independentes e, empregando a média, desvio padrão, curtose e assimetria dos valores das entropias para criar mapeamentos compartilhados. As acurácias encontradas utilizando a abordagem são apresentadas na Tabela 5.5 e, com exceção do conjunto de dados Cifar10, as melhores médias de acurácia superam a dos mapeamentos aleatórios.

Tabela 5.5: Resultados do mapeamento pelo agrupamento da entropia. EI - Entropia Independente. ECM - Entropia Compartilhado (Média). ECD - Entropia Compartilhado (Desvio padrão). ECC - Entropia Compartilhado (Curtose). ECA - Entropia Compartilhado (Assimetria). MAC - Mapeamento Aleatório Compartilhado. MAI - Mapeamento Aleatório Independente.

Método **Dataset**	EI	ECM	ECD	ECC	ECA	MAC	MAI
MNIST	92.16%	94.23%	93.42%	93.88%	94.43%	93.84%	93.33%
Fashion	76.07%	82.15%	82.49%	82.46%	81.93%	82.28%	80.71%
CK+	63.11%	58.56%	57.58%	56.44%	60.61%	55.83%	56.29%
Cifar10	22.05%	28.70%	30.13%	31.28%	31.31%	33.83%	32.20%
IMDb	60.28%	67.17%	69.79%	64.36%	64.27%	64.41%	60.35%
MovieLens	69.84%	71.22%	70.71%	69.95%	69.47%	68.92%	69.26%

O desempenho dos melhores mapeamentos gerados superou as acurácias apresen-

tadas para os mapeamentos gerados de maneira aleatória, com exceção do *dataset* Cifar10. Tais resultados ficam mais interessantes ao observarmos o tempo de criação dos mapeamentos, dispostos na Tabela 5.6, uma vez que, calculadas as entropias a partir das imagens mentais, o procedimento de agrupamento tem um baixo custo computacional.

Tabela 5.6: Tempo de criação dos mapeamentos pelo agrupamento da entropia (em milissegundos). EI - Entropia Independente. ECM - Entropia Compartilhado (Média). ECD - Entropia Compartilhado (Desvio padrão). ECC - Entropia Compartilhado (Curtose). ECA - Entropia Compartilhado (Assimetria).

Método Dataset	EI	ECM	ECD	ECC	ECA
MNIST	30.22	4.28	3.63	3.81	3.66
Fashion	21.00	2.39	2.84	2.79	2.82
CK+	126.47	30.67	49.84	102.38	124.15
Cifar10	29.31	2.60	4.06	5.96	4.42
IMDb	65.86	71.33	81.61	9.40	9.62
MovieLens	25.54	33.80	30.86	9.48	7.64

A técnica não alcança bons resultados nos conjuntos de dados que não possuam muitas similaridades entre os exemplos de mesma categoria, já que o conhecimento contido nas imagens mentais apresentará o que foi aprendido pela rede mas com pouca representatividade das estruturas dos exemplos em si. Esse é o caso do *dataset* Cifar10, no qual as acurácias foram inferiores com a técnica proposta do que ao utilizar mapeamentos aleatórios.

Para cada conjunto de dados temos que criar a imagem mental e, a partir dela, calcular a imagem de entropia. Esse processo só precisa ser feito uma vez para cada *dataset* binarizado, com o tamanho da tupla não interferindo no resultado. Desta forma, a Tabela 5.7 apresenta os tempos de criação de tais imagens de entropia e, como esses procedimentos são preliminares a criação dos mapeamentos, estes tempos não foram levados em conta nos valores apresentados na Tabela 5.6.

Tabela 5.7: Tempo de criação das imagens mentais e cálculo das entropias (em milissegundos).

Dataset	**Tempo (ms)**	
	média	**d. padrão**
MNIST	163.72	2.21
Fashion	192.53	5.13
CK+	170.58	24.60
Cifar10	155.86	20.57
IMDb	294.79	0.00
MovieLens	239.75	113.57

5.2.4 Mapeamento por restrições

Na abordagem dos mapeamentos criados a partir das restrições, duas regras foram elaboradas para a extração das restrições. A primeira regra, já descrita no Capítulo 4, é da "separação dos uns", que foi utilizada nos conjuntos de dados esparsos, e busca por *pixels* com valor igual a 1 nas imagens, criando restrições para evitar que tais endereços ocorram na mesma ênupla no mapeamento. A segunda regra, da "separação das linhas", foi criada para os conjuntos de imagens, em que uma janela deslizante busca por linhas horizontais ou verticais formadas por 4 *pixels* e, caso encontre, pares de restrições são criados com a combinação dos endereços envolvidos. A Figura 5.2 apresenta os padrões de linhas buscados pela segunda regra.

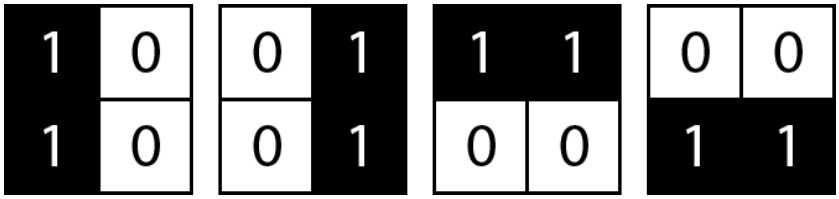


Figura 5.2: Linhas verticais e horizontais utilizadas na regra de separação.

A Tabela 5.8 apresenta as acurácias dos mapeamentos gerados. As restrições foram utilizadas diretamente para criar mapeamentos e através da união, interseção ou interseção estrita, para os mapeamentos compartilhados.

Tabela 5.8: Resultados dos mapeamentos por restrições. RI - Restrições Independentes. RCU - Restrições Compartilhadas (União). RCI - Restrições Compartilhadas (Interseção). RCE - Restrições Compartilhadas (Estrita). MAC - Mapeamento Aleatório Compartilhado. MAI - Mapeamento Aleatório Independente.

Restrições Dataset	RI	RCU	RCI	RCE	MAC	MAI
MNIST	94.35%	95.34%	93.87%	95.43%	93.84%	93.33%
Fashion	78.74%	82.15%	82.27%	81.81%	82.28%	80.71%
CK+	59.70%	55.45%	56.06%	55.76%	55.83%	56.29%
Cifar10	27.43%	30.25%	33.56%	30.45%	33.83%	32.20%
IMDb	60.43%	65.00%	61.92%	67.45%	64.41%	60.35%
MovieLens	69.52%	69.60%	69.66%	68.99%	68.92%	69.26%

Observando os resultados, podemos notar que os *datasets* Fashion e Cifar10 apresentaram resultados abaixo dos alcançados com mapeamentos aleatórios. Entretanto, os datasets esparsos tiveram resultados melhores, uma vez que a regra definida foi elaborada com base nas características dos mesmos. De forma similar, a busca por linhas também é baseada nas características do MNIST, o que justifica a melhoria no reconhecimento dos padrões do conjunto de dados. Vale destacar que nos conjuntos de dados Fashion-MNIST e Cifar-10 as melhores acurácias na abordagem de restrições tiveram um resultado inferior ao método aleatório.

É importante ressaltar que o modelo necessita de um conhecimento prévio das *features* presentes no conjunto de dados, que é descrito através da criação personalizada das regras. Desta forma, a regra definida para os *datasets* de imagens é simplista quando utilizada em padrões mais complexos, com é o caso do Cifar10 e do Fashion. A Tabela 5.9 traz os tempos de criação dos mapeamentos a partir das restrições extraídas dos conjuntos de dados binarizados.

Tabela 5.9: Tempo de criação dos mapeamentos a partir das restrições (em milissegundos). RI - Restrições Independentes. RCU - Restrições Compartilhadas (União). RCI - Restrições Compartilhadas (Interseção). RCE - Restrições Compartilhadas (Estrita).

Restrições Dataset	RI	RCU	RCI	RCE
MNIST	2518.64	175.66	143.48	175.13
Fashion	3065.10	339.80	225.39	303.03
CK+	126713.98	71060.68	17607.20	71211.89
Cifar10	8401.01	864.05	657.48	900.63
IMDb	42154.29	26037.93	7629.40	25742.61
MovieLens	73581.26	43570.41	12694.76	28536.24

O custo computacional inicial é elevado devido a extração das restrições, valores apresentados na Tabela 5.10. Entretanto, conforme descrito no Capítulo 4, o procedimento de extração só precisa ser feito uma vez para cada *dataset*, e o custo para criação de diferentes mapeamentos é baixo, independente do tamanho de tupla definido.

Tabela 5.10: Tempo de extração das restrições (em milissegundos).

Dataset	**Tempo (ms)**	
	média	**d. padrão**
MNIST	1096313.88	20703.76
Fashion	2845453.64	113500.76
CK+	33678.73	23874.53
Cifar10	3130782.79	82217.68
IMDb	369516.80	182818.64
MovieLens	98550.13	0.00

5.3 Discussão

Como é possível observar, em todos os conjuntos de dados foram encontradas uma ou mais alternativas de criação de mapeamentos para a WiSARD com melhores acurácias do que a abordagem aleatória. A Tabela 5.11 dispõe de forma resumida a melhor técnica encontrada em cada *dataset*.

Tabela 5.11: Resumo dos melhores mapeamentos por conjunto de dados. RCE - Restrições Compartilhadas (Estrita). BEA - Busca Estocástica (Aprimorado). EI - Entropia Independente. ECD - Entropia Compartilhado (Desvio padrão). AG - Algortimo Genético (Girodano).

Dataset	**Mapeamento**	**Acurácia (%)**	
		média	**d. padrão**
MNIST	RCE	95.43	0.16
Fashion	BEA	82.84	0.28
CK+	EI	63.11	1.98
Cifar10	BEA	33.91	0.43
IMDb	ECD	69.79	0.06
MovieLens	AG	71.61	0.43

Os modelos de otimização estocástica se mostraram eficazes em *datasets* com características mais complexas, ou até mesmo desconhecidas. Podemos destacar o modelo aprimorado de busca estocástica nos conjuntos de dados Fashion e Cifar10. Além disto, temos o algoritmo genético proposto por Giordano no MovieLens, que foi a única técnica não inédita a alcançar a melhor acurácia em um *dataset* no presente trabalho.

Os resultados alcançados com as técnicas de agrupamento pela entropia mostram sua eficácia, com a aplicação nos conjuntos de dados CK+ e IMDb sendo destacadas como os mapeamentos que resultaram nas melhores acurácias. Entretanto, é importante ressaltar que a suposição de que cada pixel é independente dos demais não necessariamente é verdadeira em todos os casos, principalmente ao ser utilizado com conjuntos de dados de imagens, sendo considerada uma abordagem muito restritiva [43].

Utilizar os mapeamentos gerados com o método de agrupamento da entropia como população inicial das abordagens de otimização estocástica pode apresentar algumas vantagens. No melhor caso, tal mistura de técnicas pode resultar na redução do tempo de criação dos mapeamentos nas abordagens de otimização estocástica. No pior caso, teremos um conjunto de dados em que a suposição das variáveis independentes não é válida e, de acordo com os resultados dos experimentos do presente trabalho, que ainda apresentam acurácias interessantes quando comparadas com a abordagem aleatória.

Conforme descrito no Capítulo 4, o modelo de criação de mapeamentos pelas restrições depende de um conhecimento prévio das características dos dados que serão usados ser expressado. Em contraste com o agnosticismo a tal características inerente dos modelos de otimização estocástica, esse fator pode ser limitante. Entretanto, a técnica serve como um *framework* de inserção de conhecimento na criação de mapeamentos para redes WiSARD, buscando encontrar as relações que não estão

presentes em versões sumarizadas dos dados, mas nos exemplos de forma individual.

O resultado encontrado no MNIST com o mapeamento criado a partir das restrições, extraídas com a regra da "separação das linhas", demonstra o potencial do uso do conhecimento aplicado na criação dos mapeamentos. Isso foi possível uma vez que a regra descrita foi elaborada buscando uma das representações mais simples no reconhecimento de padrões, linhas verticais e horizontais, que constituem as *features* do conjunto em questão.

Capítulo 6

Conclusão

Neste trabalho, foi realizado um estudo comparativo de diferentes abordagens para a criação de mapeamentos no classificador WiSARD. Ao serem submetidos aos principais conjuntos de dados encontrados na literatura, as soluções apresentadas provaram-se eficazes em termos de acurácia e, com exceção de um *dataset*, os melhores resultados foram alcançados pelas abordagens desenvolvidas.

O processo de pesquisa envolveu a análise das formas de criação presentes na literatura do modelo, e a introdução de técnicas, que foram desenvolvidas para solucionar as deficiências encontradas nos modelos já existentes. A ausência de abordagens que atuem em escopos diferentes na utilização dos conjuntos de dados motivou a criação das técnicas de agrupamento dos endereços pela entropia e os mapeamentos por restrições, que utilizam o conhecimento dos conjuntos de dados de formas alternativas aos métodos de otimização estocástica.

Conforme observado nos resultados experimentais, não é possível determinar uma técnica adequada que funcione em todos os cenários, mas é viável definir um conjunto de alternativas que funcionam em determinados contextos, e que podem ser utilizados de acordo com a necessidade e disponibilidade de recursos. Um exemplo seria o uso dos mapeamentos criados a partir do agrupamento, que alcançam melhores resultados que os mapeamentos aleatórios e demandam um baixo custo computacional mas que, em alguns casos, tiveram acurácias inferiores que os demais experimentos, que necessitam de um maior tempo de execução.

Por fim, o trabalho deixa algumas questões a serem exploradas. A pesquisa foi direcionada a alternativas para a criação do mapeamento no modelo WiSARD, com o pré-processamento e o tamanho da ênupla não sendo discutidos. Desta forma, são necessários estudos que abordem tais características, sua relação com o modelo e as implicações e usos em conjunto com as técnicas de mapeamento apresentadas. Tal pesquisa é fundamental para uma análise e comparação adequada com outros modelos estado-da-arte nos conjuntos de dados apresentados.

Algumas ideias de possíveis melhorias e complementos surgiram durante o de-

senvolvimento da pesquisa, porém, não entraram no escopo do presente trabalho. Tais ideias de trabalhos futuros estão listadas abaixo:

- Implementação dos algoritmos em C++, utilizando técnicas de paralelismo nos modelos de otimização estocástica para acelerar os procedimentos;
- Seleção de *features* baseada na entropia calculada a partir das imagens mentais;
- Uso dos mapeamentos por agrupamento como população inicial dos algoritmos de otimização estocástica;
- Expansão das técnicas para envolver repetição de endereços e ênuplas de tamanhos distintos;
- Elaboração de novas regras para a criação de mapeamentos por restrições;
- Implementação de preferências, no sistema de restrições, indicando eventos desejáveis na criação dos mapeamentos.
- Usar diferentes métricas além da acurácia na avaliação das abordagens, como precisão, recall, F1 score.

Referências Bibliográficas

[1] M. Claesen and B. De Moor, "Hyperparameter search in machine learning," *arXiv preprint arXiv:1502.02127*, 2015.

[2] I. Aleksander, W. Thomas, and P. Bowden, "Wisard· a radical step forward in image recognition," *Sensor review*, 1984.

[3] R. J. Mitchell, J. Bishop, and P. R. Minchinton, "Optimising memory usage in n-tuple neural networks," *Mathematics and computers in simulation*, vol. 40, no. 5-6, pp. 549–563, 1996.

[4] M. Giordano and M. De Gregorio, "An evolutionary approach for optimizing weightless neural networks,"

[5] M. Feurer, J. T. Springenberg, and F. Hutter, "Using meta-learning to initialize bayesian optimization of hyperparameters," in *Proceedings of the 2014 International Conference on Meta-learning and Algorithm Selection-Volume 1201*, pp. 3–10, Citeseer, 2014.

[6] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, pp. 2546–2554, 2011.

[7] W. W. Bledsoe and I. Browning, "Pattern recognition and reading by machine," in *Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference*, pp. 225–232, 1959.

[8] W. Bledsoe and C. Bisson, "Improved memory matrices for the n-tuple pattern recognition method," *IRE Transactions on Electronic Computers*, no. 3, pp. 414–415, 1962.

[9] H. C. Carneiro, C. E. Pedreira, F. M. França, and P. M. Lima, "The exact vc dimension of the wisard n-tuple classifier," *Neural computation*, vol. 31, no. 1, pp. 176–207, 2019.

[10] N. P. Bradshaw, *An analysis of learning in weightless neural systems*. PhD thesis, University of London, 1996.

[11] B. P. Grieco, P. M. Lima, M. De Gregorio, and F. M. França, "Producing pattern examples from "mental" images," *Neurocomputing*, vol. 73, no. 7-9, pp. 1057–1064, 2010.

[12] F. França, M. De Gregorio, P. Lima, and W. de Oliveira, "Advances in weightless neural systems," in *Proceedings of the 22th European Symposium on the Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2014.

[13] B. P. Grieco, P. M. Lima, M. De Gregorio, and F. M. França, "Producing pattern examples from "mental" images," *Neurocomputing*, vol. 73, no. 7-9, pp. 1057–1064, 2010.

[14] M. De Gregorio, "On the reversibility of multidiscriminator systems," tech. rep., Technical Report Technical Report 125/97, Istituto di Cibernetica–CNR, 1997.

[15] C. Soares, C. da Silva, M. De Gregorio, and F. França, "Uma implementação em software do classificador wisard," *V Simpósio Brasileiro de Redes Neurais*, vol. 2, pp. 225–229, 1998.

[16] J. Bishop, A. Crowe, P. Michinton, and R. Mitchell, "Evolutionary learning to optimise mapping in n-tuple networks," in *IEE Colloquium on Machine Learning*, pp. 3–1, IET, 1990.

[17] H. B. Azhar and K. Dimond, "A stochastic search algorithm to optimize an n-tuple classifier by selecting its inputs," in *International Conference Image Analysis and Recognition*, pp. 556–563, Springer, 2004.

[18] M. H. B. Azhar, F. Deravi, and K. Dimond, "Criticality dispersion in swarms to optimize n-tuples," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pp. 1–8, 2008.

[19] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[20] C. I. Watson and C. L. Wilson, "Nist special database 4," *Fingerprint Database, National Institute of Standards and Technology*, vol. 17, no. 77, p. 5, 1992.

[21] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 4, pp. 1942–1948, IEEE, 1995.

[22] M. H. B. Azhar, F. Deravi, and K. R. Dimond, "Particle swarm intelligence to optimize the learning of n-tuples," *Journal of Intelligent Systems*, vol. 17, no. Supplement, pp. 169–196, 2008.

[23] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[24] D. Whitley, "A genetic algorithm tutorial," *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.

[25] T. Lin and S. Goodwin, "Csp: Definition, creation, and algorithms,"

[26] M. Ettaouil and C. Loqman, "A new optimization model for solving the constraint satisfaction problem," *Journal of Advanced Research in Computer Science*, vol. 1, no. 1, pp. 13–31, 2009.

[27] C. Ansótegui, M. L. Bonet, J. Levy, and F. Manyà, "The logic behind weighted csp," in *Proceedings of the 20th international joint conference on Artifical intelligence*, pp. 32–37, 2007.

[28] J. E. Gallardo, C. Cotta, and A. J. Fernández, "Solving weighted constraint satisfaction problems with memetic/exact hybrid algorithms," *Journal of Artificial Intelligence Research*, vol. 35, pp. 533–555, 2009.

[29] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations research*, vol. 14, no. 4, pp. 699–719, 1966.

[30] R. Dechter, "Mini-buckets: A general scheme for generating approximations in automated reasoning," in *IJCAI*, vol. 97, pp. 1297–1303, Citeseer, 1997.

[31] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[32] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database," *URL http://yann. lecun. com/exdb/mnist*, 1998.

[33] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[34] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," *online: http://www. cs. toronto. edu/kriz/cifar. html*, vol. 55, 2014.

[35] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pp. 94–101, IEEE, 2010.

[36] N. Otsu, "A thresholding selection method from gray-scale histogram," *IEEE Transactions on System, Man, and Cybernetics. v9*, pp. 62–66.

[37] A. Kappaun, K. Camargo, F. Rangel, F. Firmino, P. M. V. Lima, and J. Oliveira, "Evaluating binary encoding techniques for wisard," in *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 103–108, IEEE, 2016.

[38] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, (Portland, Oregon, USA), pp. 142–150, Association for Computational Linguistics, June 2011.

[39] T. Joachims, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization.," tech. rep., Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

[40] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[41] A. S. Lima Filho, G. P. Guarisa, L. A. Lusquino Filho, L. F. Oliveira, F. M. Franca, and P. Lima, "wisardpkg–a library for wisard-based models," *arXiv*, pp. arXiv–2005, 2020.

[42] G. P. Guarisa, "gabrielguarisa/imazero 1.0," Aug. 2020.

[43] H. Maître, *Image processing*. Wiley-IEEE Press, 2008.

[44] L. A. L. Filho, G. P. Guarisa, L. F. Oliveira, A. L. Filho, F. M. França, and P. M. Lima, "Action units classification using cluswisard,"

[45] L. A. L. Filho, L. F. Oliveira, A. L. Filho, G. P. Guarisa, P. M. Lima, and F. M. França, "Prediction of palm oil production with an enhanced n-tuple regression network,"

[46] A. L. Filho, G. P. Guarisa, L. A. L. Filho, L. F. Oliveira, C. A. Cosenza, F. M. França, and P. M. Lima, "Interpretation of model agnostic classifiers via local mental images,"

[47] L. A. Lusquino Filho, L. F. Oliveira, H. C. Carneiro, G. P. Guarisa, A. Lima Filho, F. M. França, and P. M. Lima, "A weightless regression system for predicting multi-modal empathy," in *2020 15th IEEE International*

Conference on Automatic Face and Gesture Recognition (FG 2020)(FG), pp. 554–558, 2020.

[48] L. A. Lusquino Filho, L. F. Oliveira, A. Lima Filho, G. P. Guarisa, L. M. Felix, P. M. Lima, and F. M. França, "Extending the weightless wisard classifier for regression," *Neurocomputing*, 2020.

Apêndice A

Resultados dos Experimentos

Este apêndice trata da exploração dos resultados dos experimentos com a criação dos mapeamentos para o modelo WiSARD utilizando os *datasets* apresentados no Capítulo 5.

A média e o desvio padrão da acurácia e do tempo de criação dos mapeamentos são apresentados para cada tipo de experimento proposto no trabalho, de acordo com a configuração de binarização e tamanho da tupla que resultam na melhor acurácia média. A escolha de apresentar os dados dessa forma foi tomada buscando uma sintetização dos resultados alcançados nos experimentos, uma vez que expor todos os valores para todas as combinações seria inviável.

A.1 Resultados

Tabela A.1: Resultados utilizando mapeamentos aleatórios - Compartilhado.

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	28	93.60	0.23	0.08	0.01
	Otsu	28	92.62	0.30	0.10	0.10
	Valor Médio	28	93.84	0.18	0.08	0.02
Fashion	Local	28	80.80	0.25	0.09	0.02
	Otsu	28	81.65	0.20	0.09	0.02
	Valor Médio	28	82.28	0.29	0.08	0.02
CK+	Local	10	49.85	2.35	0.29	0.07
	Otsu	10	53.18	1.09	0.27	0.04
	Valor Médio	10	55.83	1.23	0.25	0.04
Cifar10	Local	16	25.59	0.34	0.09	0.02
	Otsu	16	33.83	0.34	0.09	0.02
	Valor Médio	16	32.12	0.37	0.09	0.02
IMDb	Termômetro	29	64.41	0.28	0.24	0.12
MovieLens	Termômetro	58	68.92	1.40	0.19	0.02

Tabela A.2: Resultados utilizando mapeamentos aleatórios - Independente.

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	28	92.90	0.42	1.25	0.20
	Otsu	28	91.77	0.46	1.23	0.17
	Valor Médio	28	93.33	0.25	1.26	0.28
Fashion	Local	28	79.86	0.27	1.05	0.02
	Otsu	28	80.17	0.38	1.09	0.06
	Valor Médio	28	80.71	0.27	1.06	0.02
CK+	Local	10	49.17	2.22	9.80	0.45
	Otsu	10	53.03	1.55	1.58	0.28
	Valor Médio	16	56.29	2.42	1.36	0.06
Cifar10	Local	16	25.28	0.43	1.71	0.37
	Otsu	16	32.20	0.28	0.23	0.03
	Valor Médio	16	31.08	0.34	0.25	0.05
IMDb	Termômetro	55	60.35	0.35	1.02	0.06
MovieLens	Termômetro	29	69.26	1.12	1.07	0.02

Tabela A.3: Resultados dos experimentos de otimização estocástica - PSO (Azhar).

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	28	92.24	1.02	260249.27	23190.64
	Otsu	28	90.85	0.82	328492.78	28510.79
	Valor Médio	28	92.37	0.75	327307.58	33323.32
Fashion	Local	16	76.04	0.55	426970.38	21137.00
	Otsu	28	76.71	0.94	233513.13	25202.08
	Valor Médio	28	78.05	0.84	264524.26	21011.35
CK+	Local	4	53.56	2.47	364816.27	10919.56
	Otsu	8	57.50	2.17	158638.41	8462.68
	Valor Médio	8	61.14	1.50	148318.58	7001.96
Cifar10	Local	16	22.64	0.57	738753.84	18582.46
	Otsu	16	30.73	0.61	669417.50	31258.17
	Valor Médio	16	29.74	0.50	834980.25	15393.07
IMDb	Termômetro	5	62.21	0.49	556508.75	95802.71
MovieLens	Termômetro	58	69.76	1.96	39125.70	4702.59

Tabela A.4: Resultados dos experimentos de otimização estocástica - Busca estocástica (Azhar).

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	28	92.47	0.60	40913.94	4449.58
	Otsu	28	91.26	0.74	41194.14	4171.64
	Valor Médio	28	92.89	0.43	42804.63	2832.21
Fashion	Local	28	80.20	0.36	62944.90	6329.98
	Otsu	28	80.63	0.37	43509.89	3262.53
	Valor Médio	28	81.18	0.41	46435.27	3852.59
CK+	Local	4	52.65	1.46	66734.95	4148.51
	Otsu	4	55.08	1.92	56604.76	1932.93
	Valor Médio	8	57.35	1.58	40099.80	3673.54
Cifar10	Local	16	25.98	0.37	247407.19	11984.34
	Otsu	16	32.83	0.36	145574.28	5869.48
	Valor Médio	16	31.64	0.40	204440.13	11456.46
IMDb	Termômetro	25	64.83	0.43	50353.75	4439.54
MovieLens	Termômetro	5	69.52	2.04	39584.86	4971.87

Tabela A.5: Resultados dos experimentos de otimização estocástica - Busca estocástica (Aprimorado).

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	28	94.96	0.19	51969.76	3218.84
	Otsu	28	94.37	0.20	51295.20	4342.28
	Valor Médio	28	95.08	0.15	52213.80	3866.82
Fashion	Local	28	81.46	0.28	45388.14	2176.89
	Otsu	28	81.95	0.29	39777.06	1760.64
	Valor Médio	28	82.84	0.28	46590.22	2623.53
CK+	Local	5	58.71	1.46	27837.33	1145.16
	Otsu	5	56.59	1.13	23103.44	1156.89
	Valor Médio	4	61.29	1.25	24880.13	1938.02
Cifar10	Local	16	25.04	0.38	95878.96	4670.55
	Otsu	16	33.91	0.43	83671.84	4000.91
	Valor Médio	16	32.52	0.28	89510.02	3063.33
IMDb	Termômetro	29	68.59	0.45	127885.04	4985.51
MovieLens	Termômetro	29	68.52	1.27	3823.64	444.78

Tabela A.6: Resultados dos experimentos de otimização estocástica - Algoritmo genético (Giordano).

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	28	93.79	0.37	210884.71	56442.08
	Otsu	28	92.89	0.30	234994.24	51598.19
	Valor Médio	28	94.00	0.28	205192.98	65440.62
Fashion	Local	28	81.42	0.39	221640.12	42411.42
	Otsu	28	81.30	0.33	228732.31	61714.64
	Valor Médio	28	81.95	0.39	569841.03	138148.94
CK+	Local	25	51.97	1.64	5748.47	2780.11
	Otsu	50	51.97	2.36	3396.96	1740.63
	Valor Médio	50	53.86	2.11	2522.83	1316.32
Cifar10	Local	16	25.29	0.49	1493479.27	307576.97
	Otsu	16	30.62	0.70	1523520.83	312068.81
	Valor Médio	16	29.81	0.51	708142.97	49489.60
IMDb	Termômetro	5	64.02	0.02	130532.24	50726.38
MovieLens	Termômetro	58	71.61	0.43	2220.18	1070.34

Tabela A.7: Resultados dos experimentos de otimização estocástica - Algoritmo genético (Aprimorado).

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	28	93.93	0.36	184550.65	78411.97
	Otsu	28	93.05	0.20	156662.89	74834.63
	Valor Médio	28	94.17	0.28	147724.87	58305.96
Fashion	Local	28	81.17	0.31	210905.38	69603.58
	Otsu	28	81.69	0.29	225546.89	86708.11
	Valor Médio	28	82.29	0.30	218820.77	94762.04
CK+	Local	5	48.03	1.71	21844.24	7024.12
	Otsu	10	53.33	1.05	13609.16	5439.36
	Valor Médio	10	55.83	1.72	15638.49	5710.63
Cifar10	Local	16	25.56	0.37	719291.88	272338.05
	Otsu	16	33.59	0.33	492738.26	175187.99
	Valor Médio	16	32.25	0.41	478582.38	200744.18
IMDb	Termômetro	29	64.34	0.36	211406.33	57909.21
MovieLens	Termômetro	58	69.39	2.00	5829.13	1622.59

Tabela A.8: Resultados do mapeamento pelo agrupamento da entropia - Independente.

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	16	90.40	0.08	22.14	3.81
	Otsu	16	88.28	0.08	29.25	7.75
	Valor Médio	16	92.16	0.05	30.22	7.85
Fashion	Local	28	76.05	0.06	26.13	5.66
	Otsu	28	76.07	0.09	21.00	3.70
	Valor Médio	28	75.54	0.08	25.48	4.76
CK+	Local	16	63.11	1.98	126.47	18.53
	Otsu	25	59.02	1.43	76.81	7.13
	Valor Médio	16	59.85	1.52	83.57	19.15
Cifar10	Local	16	22.05	0.23	29.31	4.61
	Otsu	16	19.14	0.20	28.54	6.55
	Valor Médio	16	20.39	0.22	28.31	5.71
IMDb	Termômetro	55	60.28	0.45	65.86	13.74
MovieLens	Termômetro	58	69.84	1.19	25.54	1.73

Tabela A.9: Resultados do mapeamento pelo agrupamento da entropia - Média.

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	16	94.05	0.07	3.46	0.57
	Otsu	16	93.57	0.03	3.78	0.56
	Valor Médio	16	94.23	0.04	4.28	1.49
Fashion	Local	28	80.81	0.11	2.41	0.55
	Otsu	28	81.16	0.11	2.36	0.18
	Valor Médio	28	82.15	0.11	2.39	0.49
CK+	Local	10	52.20	1.87	18.37	2.27
	Otsu	25	57.95	1.90	31.84	4.69
	Valor Médio	20	58.56	0.74	30.67	7.45
Cifar10	Local	16	25.94	0.32	2.43	0.87
	Otsu	16	28.70	0.32	2.60	0.57
	Valor Médio	16	27.96	0.15	2.78	0.72
IMDb	Termômetro	29	67.17	0.09	71.33	10.92
MovieLens	Termômetro	58	71.22	0.65	33.80	2.86

Tabela A.10: Resultados do mapeamento pelo agrupamento da entropia - Desvio padrão.

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	28	93.10	0.05	3.49	0.60
	Otsu	28	92.48	0.08	3.52	0.70
	Valor Médio	28	93.42	0.04	3.63	1.42
Fashion	Local	28	81.35	0.08	2.44	0.62
	Otsu	28	81.64	0.15	2.64	0.58
	Valor Médio	28	82.49	0.08	2.84	0.89
CK+	Local	16	55.98	2.17	81.74	17.77
	Otsu	16	54.70	1.09	57.71	16.67
	Valor Médio	10	57.58	1.47	49.84	9.67
Cifar10	Local	16	25.82	0.24	3.42	0.71
	Otsu	16	30.13	0.22	4.06	1.58
	Valor Médio	16	29.94	0.15	3.19	1.16
IMDb	Termômetro	25	69.79	0.06	81.61	17.22
MovieLens	Termômetro	58	70.71	0.86	30.86	1.85

Tabela A.11: Resultados do mapeamento pelo agrupamento da entropia - Curtose.

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	28	93.88	0.00	3.81	1.68
	Otsu	28	92.69	0.00	2.72	0.22
	Valor Médio	28	93.75	0.06	2.66	0.20
Fashion	Local	28	80.49	0.08	2.46	0.46
	Otsu	28	81.78	0.10	3.35	0.95
	Valor Médio	28	82.46	0.08	2.79	0.46
CK+	Local	8	48.56	1.87	124.00	17.01
	Otsu	20	55.30	1.43	109.82	22.20
	Valor Médio	20	56.44	1.83	102.38	12.37
Cifar10	Local	16	26.04	0.24	3.48	0.56
	Otsu	16	31.28	0.18	5.96	9.04
	Valor Médio	16	31.19	0.16	3.68	0.69
IMDb	Termômetro	29	64.36	0.34	9.40	1.78
MovieLens	Termômetro	58	69.95	1.69	9.48	0.67

Tabela A.12: Resultados do mapeamento pelo agrupamento da entropia - Assimetria.

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	16	94.14	0.00	2.71	0.39
	Otsu	16	93.45	0.00	3.32	0.34
	Valor Médio	16	94.43	0.00	3.66	0.50
Fashion	Local	16	81.25	0.12	2.94	0.82
	Otsu	28	81.76	0.00	2.93	0.65
	Valor Médio	28	81.93	0.01	2.82	0.72
CK+	Local	4	48.79	1.05	157.45	29.00
	Otsu	20	56.74	1.34	156.94	34.03
	Valor Médio	25	60.61	1.30	124.15	15.48
Cifar10	Local	16	26.10	0.15	4.61	1.53
	Otsu	16	31.31	0.02	4.42	1.08
	Valor Médio	16	30.10	0.13	4.33	0.83
IMDb	Termômetro	29	64.27	0.31	9.62	2.00
MovieLens	Termômetro	58	69.47	1.18	7.64	0.69

Tabela A.13: Resultados dos mapeamentos por restrições - Independente.

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	16	93.62	0.25	2148.09	416.59
	Otsu	16	91.95	0.28	1748.02	23.86
	Valor Médio	16	94.35	0.15	2518.64	121.99
Fashion	Local	28	77.37	0.32	3518.78	564.44
	Otsu	28	77.17	0.38	3131.75	625.40
	Valor Médio	28	78.74	0.34	3065.10	894.65
CK+	Local	16	59.70	2.75	126713.98	647.96
	Otsu	16	54.70	1.47	125844.46	990.64
	Valor Médio	20	56.74	1.25	104044.78	171.23
Cifar10	Local	16	24.63	0.31	7005.10	2314.39
	Otsu	16	27.20	0.40	8025.48	1963.41
	Valor Médio	16	27.43	0.35	8401.01	1929.45
IMDb	Termômetro	55	60.43	0.35	42154.29	1012.53
MovieLens	Termômetro	58	69.52	1.57	73581.26	1645.24

Tabela A.14: Resultados dos mapeamentos por restrições - União.

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	16	95.14	0.12	171.06	10.04
	Otsu	16	94.53	0.17	176.48	10.87
	Valor Médio	16	95.34	0.18	175.66	12.49
Fashion	Local	28	81.07	0.24	324.22	105.27
	Otsu	28	81.42	0.29	310.92	67.24
	Valor Médio	28	82.15	0.32	339.80	65.64
CK+	Local	10	49.24	2.28	28565.99	137.38
	Otsu	5	53.64	1.99	57174.71	252.17
	Valor Médio	4	55.45	1.24	71060.68	334.17
Cifar10	Local	16	26.99	0.36	849.17	148.54
	Otsu	16	30.25	0.36	864.05	57.49
	Valor Médio	16	30.24	0.35	694.52	194.37
IMDb	Termômetro	55	65.00	0.32	26037.93	138.12
MovieLens	Termômetro	58	69.60	1.33	43570.41	2166.06

Tabela A.15: Resultados dos mapeamentos por restrições - Interseção.

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	28	93.59	0.23	104.04	9.94
	Otsu	28	92.73	0.35	128.56	11.88
	Valor Médio	28	93.87	0.25	143.48	12.27
Fashion	Local	28	80.84	0.30	229.81	45.47
	Otsu	28	81.61	0.29	297.54	38.09
	Valor Médio	28	82.27	0.26	225.39	60.36
CK+	Local	8	48.86	2.02	35765.52	359.82
	Otsu	10	52.88	1.09	28466.37	131.93
	Valor Médio	16	56.06	1.39	17607.20	90.39
Cifar10	Local	16	25.56	0.31	532.82	124.89
	Otsu	16	33.56	0.34	657.48	128.67
	Valor Médio	16	32.25	0.28	770.13	192.09
IMDb	Termômetro	29	61.92	0.38	7629.40	52.07
MovieLens	Termômetro	29	69.66	0.73	12694.76	1153.94

Tabela A.16: Resultados dos mapeamentos por restrições - Estrita.

Dataset	**Binarização**	**n**	**Acurácia (%)**		**Tempo (ms)**	
			média	**d. padrão**	**média**	**d. padrão**
MNIST	Local	16	95.10	0.13	188.54	28.07
	Otsu	16	94.50	0.17	208.53	27.50
	Valor Médio	16	95.43	0.16	175.13	14.85
Fashion	Local	28	81.03	0.23	280.89	86.47
	Otsu	28	80.62	0.28	379.77	47.14
	Valor Médio	28	81.81	0.23	303.03	71.45
CK+	Local	10	49.39	2.53	28677.96	157.98
	Otsu	4	53.64	1.51	71244.98	284.99
	Valor Médio	4	55.76	1.36	71211.89	306.10
Cifar10	Local	16	26.86	0.41	805.96	230.91
	Otsu	16	30.45	0.40	900.63	150.68
	Valor Médio	16	30.13	0.36	922.82	208.98
IMDb	Termômetro	25	67.45	0.27	25742.61	125.98
MovieLens	Termômetro	58	68.99	1.27	28536.24	1100.84

Apêndice B

Trabalhos aceitos

Neste apêndice estão inclusas as primeiras páginas dos trabalhos desenvolvidos em co-autoria durante a preparação da atual dissertação.

Em [44], os modelos WiSARD e ClusWiSARD foram usados na classificação de *Action Units*, que indicam os músculos faciais envolvidos em expressões de emoções. No trabalho desenvolvido em [45], os modelos de regressão RegressionWiSARD e ClusRegressionWiSARD são apresentados em um problema de predição da produção de óleo de palmeiras. Já em [46], um interpretador de classificadores é introduzido, utilizando imagens mentais locais em seu funcionamento. Os três artigos descritos anteriormente foram publicados em anais de congressos.

O trabalho [47] é um resumo expandido, no qual os modelos RegressionWiSARD e ClusRegressionWiSARD são usados para predizer a valência de emoções.

Em [48], os modelos RegressionWiSARD e ClusRegressionWiSARD são detalhados e definidos com uma extensão do classificador WiSARD. Tal trabalho foi publicado como artigo completo em um periódico.

Por fim, temos [41], que apresenta a biblioteca wisardpkg com os modelos de classificação, WiSARD e ClusWiSARD, e regressão, RegressionWiSARD e ClusRegressionWiSARD. Biblioteca esta que foi utilizada no presente trabalho.

Action Units Classification using ClusWiSARD

Leopoldo A. D. Lusquino Filho[1]*, Gabriel P. Guarisa[1], Luiz F. R. Oliveira[1], Aluizio Lima Filho[1], Felipe M. G. França[1], and Priscila M. V. Lima[2]

1- PESC/COPPE 2- NCE
Universidade Federal do Rio de Janeiro, RJ, Brazil **
lusquino@cos.ufrj.br

Abstract. This paper presents the use of WiSARD and ClusWiSARD weightless neural networks models for the classification of the contraction and extension of *Action Units*, the facial muscles involved in emotive expressions. This is a complex problem due to the large number of very similar classes, and because it is a multi-label classification task, where the positive expression of one class can modify the response of the others. WiSARD and ClusWiSARD solutions are proposed and validated using the CK+ dataset, producing responses with accuracy of 89.66%. Some of the major works in the field are cited here, but a proper comparison is not possible due to a lack of appropriate information about such solutions, such as the subset of classes used and the time of training/testing. The contribution of this paper is in the pioneering use of weightless neural networks in an AUs classification task, in the unpublished application of the WiSARD and ClusWiSARD models in multi-label tasks and in the new unsupervised expansion of ClusWiSARD proposed here.

Keywords: Action Units, WiSARD, ClusWISARD, weightless neural network

1 Introduction

Ekman and Friesen [1] cataloged a set of muscles known as *Action Units* (AUs) – which would be responsible for all facial expressiveness – while attempting to obtain a set of universal emotions present in any human. The automatic identification of these AUs has been developed since the mid-1990s and has several applications: forensics, psychological treatment, physical therapy support and advertising feedback, among others. AUs have also been used in the development of adaptive digital avatars [2].

Some of the great difficulties in automatic detection of AUs are the large number of classes and the wide variety of forms how AUs express themselves, besides the fact that they usually manifest together, making this a hard multi-label task. In this way, the approaches that are emerging in the literature usually

* Corresponding author

** This work was partially supported by CAPES, CNPq, FAPERJ and FINEP, Brazilian research agencies.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

Prediction of Palm Oil Production with an Enhanced n-Tuple Regression Network

Leopoldo A. D. Lusquino Filho[1], Luiz F. R. Oliveira[1], Aluizio L. Filho[1], Gabriel P. Guarisa[1], Priscila M. V. Lima[2], Felipe M. G. França[1] *

1- PESC/COPPE 2- NCE
Universidade Federal do Rio de Janeiro, RJ, Brazil

Abstract. This paper introduces Regression WiSARD and ClusRegression WiSARD, two new weightless neural network models that were applied in the challenging task of predicting the total palm oil production of a set of 28 differently located sites under different climate and soil profiles. Both models were derived from the n-tuple regression weightless neural model and obtained error (MAE) rates of 0.08737% and 0.08938%, respectively, which are very competitive with the state-of-art (0.07569), whilst being four (4) orders of magnitude faster during the training phase.

1 Introduction

Regression is one of the most important machine learning tasks, given the wide range of practical situations in the real world where it is necessary to predict values in a given continuum space. Due to its great utility, it is desirable that simple devices, such as small sensors, could perform regression with online training. Weightless artificial neural networks (WANNs), due to its lean, RAM-based architecture, seems to be ideal for this type of task.

This paper presents and explores the use of WANNs in the KDD18 competition [5], a challenge which goal is to predict the palm oil harvest productivity of a set of 28 different production fields using data provided by an agribusiness company. The dataset contains information about palm trees varieties, harvest dates, atmospheric data during the development of the trees, and soil characteristics of the fields where the trees are located in. The novel WANN models are based on the n-tuple Regression Network [3], which has been proved successful when compared to other classical regression approaches in non-linear plant approximation, and Mackey-Glass chaotic time series prediction tasks.

The remainder of this text is organized as follows: Section 2 presents the two weightless models proposed for regression, as well as the basic concepts behind the models that inspired it: WiSARD [1] and n-tuple Regression Network. Section 3 discusses the various approaches used in the KDD18 competition, as well as a comparison with state-of-the-art methods and other relevant results. Conclusion and future work are presented in Section 4.

*This work was partially supported by CAPES, CNPq, FAPERJ and FINEP, Brazilian research agencies.

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

Interpretation of Model Agnostic Classifiers via Local Mental Images

Aluizio Lima Filho[1], Gabriel P. Guarisa[1], Leopoldo A.D. Lusquino Filho[1], Luiz F. R. Oliveira[1], Carlos A. N. Cosenza[3], Felipe M. G. França[1] and Priscila M. V. Lima[1,2] *

1–PESC/COPPE, 2–NCE, 3–PEP/COPPE
Universidade Federal do Rio de Janeiro, RJ, Brazil

Abstract. Although successful black-box learning models have been created, understanding what happens when a machine produces a classification response is still a challenge. This work introduces FRWI – Fuzzy Regression WiSARD Interpreter, a novel fuzzy rules-based algorithm that is capable of interpreting the responses of black-box classifiers via the production of local mental images from a WiSARD n-tuple classifier. FRWI is compared with LIME – Local Interpretable Model-Agnostic Explanations, a pioneering agnostic classification interpreter model. To make a quantitative evaluation of interpretable models, a new metric – Interpretation Capacity Score – is proposed. Using this metric, it is shown that FRWI surpasses LIME in producing coherent interpretations.

1 Introduction

The need to interpret responses from learning models gets higher in different situations [1]. Questions arise such as: how the models make the decision in the classification, or when to trust its process, and when not to do so. One way to answer the first question is to show what is relevant to the model. LIME [2] – Local Interpretable Model-Agnostic Explanations – was developed with the motivation to clarify such relevance. There are other interpreter models focused on DNNs, like Gran-Cam [3], that were later introduced in the literature. How ever, LIME does not have feasible interpretation capacity for all learning models, due to interpretable models have scenarios where they work better as learning models. Experimental tests were performed utilizing LIME to explain decisions made by following classifiers: WiSARD [4], Linear model [5] and Random Forest model [6] trained with images data sets. It will be shown that results will select too much in the image as relevant, and it will not let it clear what is happening inside the classifier.For that reason, the idea of creating a degree of relevance for each pixel in the image came as an alternative to interpret the responses of black-box classifiers more feasible. This work introduces FRWI – Fuzzy Regression WiSARD Interpreter, a WiSARD n-tuple classifier that produces local mental images, via a fuzzy rules-based algorithm, as an interpretation of the responses of black-box classifiers. To compare the interpretation capacity of both LIME and FRWI models, the Interpretation Capacity Score metric is defined.

*This work was partially supported by NGD Systems, Inc./COPPETEC grant PESC21713; CAPES, CNPq and FAPERJ, Brazilian research agencies.

A weightless regression system for predicting multi-modal empathy

Leopoldo A. D. Lusquino Filho[1], Luiz F. R. Oliveira[1], Hugo C. C. Carneiro[1], Gabriel P. Guarisa[1], Aluízio Lima Filho[1], Felipe M. G. França[1] and Priscila M. V. Lima[1 2]
[1] PESC/COPPE, [2] NCE - Universidade Federal do Rio de Janeiro, RJ, Brazil

***Abstract*— This work takes into account the benefits of machine learning in order to estimate the valence of emotions on the OMG Empathy dataset, considering the information obtained from face expressions and dialogue of interlocutors. RegressionWiSARD and ClusRegressionWiSARD n-tuple regressors and its ensembles were employed to this end. The best performance achieved among all the combinations of weightless neural models considered (evaluated using the CCC metric) was 0.25 in validation set of the Personalized Track .**

I. Introduction

Since emotional states are a fundamental part of the core of human psychology, often exceeding the intellect itself in psychological hierarchy, it is natural that Affective Computing[11] occupies a prominent place in the study of the human-machine interface. Along with the apogee of machine learning, Affective Computing has experienced great growth in recent years, but it still has many open questions. Some of the main ones involve the prediction of emotions based on information from many different sources and the identification of subtle emotional states in real time. Specifically, many advances have been made recently in the area of affective prediction[25][26][27][28][29][30][31][32].

In order to offer a significant contribution in the area, this paper discusses the use of weightless neural network ensembles in predicting the affective valence of individuals in conversation videos, since this type of model has computational simplicity, great computational agility and ease of being parallelized.

The structure of this work is as follows: in Section 2 the WiSARD weightless model, some of its extensions and ensembles will be described, Section 3 deals with the preprocessing of data from different sources, Section 4 describes the experiments carried out with different types of ensembles using uni and multimodal data and discusses their results, and Section 5 is the conclusion of this work, summarizing all the previous discussion and also offering the main ongoing works.

II. n-Tuple models

The n-Tuple classifier is a boolean node pattern classifier [3], which distances itself from models derived from perceptron because it do not use synaptic weights between their neurons, thus avoiding all training time required for their convergence. n-Tuple classifier does not need any parameter fine tuning, nor does it use any error minimization technique to obtain generalization in pattern learning [17]. The family of models derived from the n-Tuple classifier is known as Weightless Artificial Neural Network (WANN).

This work was not supported by any organization

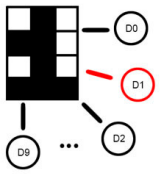


Fig. 1. The WiSARD model multidiscriminator structure. For digits recognition task there are ten discriminators. In the training phase, only the corresponding discriminator is accessed.

A. WiSARD

WiSARD is a neural model based on the n-tuple classifier, where each neuron is equivalent to a piece of memory [1]. This model is class discriminator-oriented, where all discriminators are formed by N RAM-neurons, whose memory addresses are addressed by n-bit tuples. Each neuron has 2^n memory locations.

WiSARD works with binary standards, requiring the use of some preprocessing technique to form data suitable to the model before the training and classification process. The training process consists of using the binary input to access specific memory positions of the corresponding discriminator and increment the counter that constitutes its content. During the classification, all discriminators are accessed and they are assigned a score formed by the number of non-null positions accessed. The discriminator with the highest score will determine the class of the entry and in case of a tie, a threshold called bleaching, which is initialized to zero, is increased and the classification is repeated, considering for the score only memory locations whose counter has higher value than bleaching. This procedure is repeated until there is a winning discriminator or until the bleaching value exceeds the highest counter among the memory locations accessed, in which case a default class is chosen for the entry. The structure and the training process in WiSARD are illustrated in Figs. 1 and 2. WiSARD can be used to accelerate the training of deep models, and can be used as a starting layer for such neural networks in a hybrid hierarchy[33].

Neurocomputing xxx (xxxx) xxx

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Extending the weightless WiSARD classifier for regression

Leopoldo A.D. Lusquino Filho[a,1,*], Luiz F.R. Oliveira[a,1], Aluizio Lima Filho[a], Gabriel P. Guarisa[a], Lucca M. Felix[b], Priscila M.V. Lima[a,c], Felipe M.G. França[a]

[a] PESC/COPPE, Universidade Federal do Rio de Janeiro, RJ, Brazil
[b] DCC, Universidade Federal do Rio de Janeiro, RJ, Brazil
[c] NCE, Universidade Federal do Rio de Janeiro, RJ, Brazil

ARTICLE INFO

Article history:
Received 15 July 2019
Revised 11 December 2019
Accepted 12 December 2019
Available online xxx

Communicated by Dr. Oneto Luca

Keywords:
Regression WiSARD
WiSARD
ClusWiSARD
n-Tuple classifier
n-Tuple Regression Network
Ensemble
Online learning

ABSTRACT

This paper explores two new weightless neural network models, Regression WiSARD and ClusRegression WiSARD, in the challenging task of predicting the total palm oil production of a set of 28 (twenty eight) differently located sites under different climate and soil profiles. Both models were derived from Kolcz and Allinson's *n*-Tuple Regression weightless neural model and obtained mean absolute error (MAE) rates of 0.09097 and 0.09173, respectively. Such results are very competitive with the state-of-the-art (0.07983), whilst being four orders of magnitude faster during the training phase. Additionally the models have been tested on three classic regression datasets, also presenting competitive performance with respect to other models often used in this type of task.

1. Introduction

Regression is a traditional and important machine learning task, since there is a wide range of practical situations in the real world where it is necessary to predict values in a continuous space. In a precision agriculture scenario, it would be desirable that simple devices, such as small sensors, could perform regression. Weightless Artificial Neural Networks (WANNs), due to its lean, RAM-based architecture, seem to be a suitable computational intelligence model for this type of task.

This paper explores the use of WANNs in the KDD18 competition [3], a challenge which goal is to predict the palm oil harvest productivity of a set of 28 (twenty eight) different production fields using data provided by an agribusiness company. The dataset contains information about palm trees varieties, harvest dates, atmospheric data during the development of the trees, and soil characteristics of the fields where the trees are located in. The WANN models explored in this work are based on the *n*-Tuple Regression Network [2], which was proved to be successful when compared to other classical regression approaches in non-linear plant approximation [33] and Mackey-Glass chaotic time series prediction tasks [32]. These WANN models were introduced in [5]. Here, a wider theoretical background is presented, alongside a broader exploration of their parameters and how the models perform when combined as ensembles.

The remainder of this text is organized as follows. Section 2 presents the basic models that inspired the new weightless regression ones: *n*-Tuple Classifier, WiSARD [1], ClusWISARD [8], and *n*-Tuple Regression Network [2]. Section 3 presents the two weightless models proposed for regression, and the ensemble techniques explored. Section 4 discusses the various approaches used in the KDD18 competition, as well as a comparison with state-of-the-art methods. This section also contains the description of experiments using the new models in the House Prices, CalCOFI, and Parkinson datasets. Concluding remarks and ongoing work are presented in Section 5.

2. *n*-Tuple Classifier and family

2.1. n-Tuple Classifier

The *n*-Tuple Classifier is a binary pattern classifier [4] based on Random Access Memories (RAMs), requiring no parameter fine tuning or any error minimization technique to achieve generalized learning patterns [34,35]. The basis of its operation is to use the

* Corresponding author.
E-mail address: lusquino@cos.ufrj.br (L.A.D. Lusquino Filho).
[1] Both authors had equal participation and are first author.

https://doi.org/10.1016/j.neucom.2019.12.134
0925-2312/

wisardpkg - A library for WiSARD-based models

Aluízio S. Lima Filho[1], Gabriel P. Guarisa[1], Leopoldo A. D. Lusquino Filho[1], Luiz F. R. Oliveira[1], Felipe M. G. França[1], Priscila M. V. Lima[1] [2]
[1] PESC/COPPE/UFRJ, [2] NCE/UFRJ

May 5, 2020

Abstract

In order to facilitate the production of codes using WiSARD-based models, LabZero developed an ML library C++/Python called wisardpkg. This library is an MIT-licensed open-source package hosted on GitHub under the license.

1 Introduction

Weightless artificial neural networks (WANN) are neural models that do not use weighted synapses to store the information it learns from presented patterns. Alternatively, it possesses RAM (random-access-memory)-based neurons in which information storage takes place. In a WANN, learning of a pattern corresponds to writing in memory, whereas classification essentially corresponds to the reading of certain memory positions. The advantages of these models lie essentially in their speed, simplicity, low computational and power costs and, above all, in their ability to perform online learning.

The WiSARD (Wilkes, Stonham and Aleksander Recognition Device) [1] is a pioneering WANN model that was originally designed to solve simple classification tasks. WiSARD has received extensions to perform semi-supervised and unsupervised learning, regression tasks, and improvements in training and classification policies.

Recent applications of WiSARD with remarkable results corroborating the choice of this model: data stream clustering [4, 5, 6]; time-series classification [7]; audio processing [8]; online tracking of objects [9]; GPS trajectory classification [10]; part-of-speech tagging [11, 12]; text categorization [14]; hardware assisted security [15]; emotional analysis [16, 17, 18, 19], and; prediction tasks [3, 20]. Also, recent studies on theoretical aspects of the model can be found in [13].

To facilitate the application of WiSARD and its extensions, a library in C/C++ was created, with a wrapper for Python, called wisardpkg. Section 02 of this work describes the models present in wisardpkg, while Section 03 gives details of their implementation, use, configurations and installation.