COMPARAÇÃO DE DESEMPENHO ENTRE OS MODELOS NEURAIS ÁGEIS ELM E WISARD

Luiz Fernando dos Reis de Oliveira

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.

Orientador: Felipe Maia Galvão França

Rio de Janeiro
Março de 2017

COMPARAÇÃO DE DESEMPENHO ENTRE OS MODELOS NEURAIS ÁGEIS ELM E WISARD

Luiz Fernando dos Reis de Oliveira

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) 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.

Examinada por:

Prof. Felipe Maia Galvão França, Ph.D.

Prof. Carlos Eduardo Pedreira, Ph.D.

Prof. Priscila Machado Vieira Lima, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2017

Oliveira, Luiz Fernando dos Reis de

Comparação de desempenho entre os modelos neurais ágeis ELM e WiSARD/Luiz Fernando dos Reis de Oliveira. – Rio de Janeiro: UFRJ/COPPE, 2017.

XII, 45 p.: il.; 29, 7cm.

Orientador: Felipe Maia Galvão França

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2017.

Referências Bibliográficas: p. 28 – 31.

1. Performance. 2. WiSARD. 3. ELM. I. França, Felipe Maia Galvão. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Ao meu avô,
que deixou mais que saudades.

Agradecimentos

O período que passei no mestrado foi, sem dúvidas, o de maior crescimento e amadurecimento da minha vida. Um período conturbado, difícil e intenso, com momentos em que a única saída parecia ser desistir. Mas consegui chegar ao fim com o apoio de várias pessoas, às quais gostaria de agradecer, pois foram fundamentais para a conclusão do trabalho.

Agradeço primeiramente aos meus pais, Edilene e Jorge, por todo o suporte em casa e pelo apoio que me deram para seguir a área acadêmica.

Agradeço à minha namorada, Laís, sem dúvida a pessoa que mais sofreu em todo esse tempo. Obrigado por ouvir minhas reclamações, meus lamentos e por ignorar tudo e me incentivar constantemente.

Agradeço à todos da minha família que estiveram presentes durante o processo, sem entender muito o que eu fazia, mas orgulhosos de onde cheguei.

Agradeço aos amigos que vieram desde a época da Rural e aos que fiz aqui no PESC, pela companhia, pelos almoços, pelos momentos de descontração e pelas conversas mais técnicas.

Agradeço ao professor Felipe França, pela orientação, confiança e, principalmente, pela paciência e pela compreensão dos momentos de dificuldade que passei.

Agradeço à CAPES pelo fomento e aos funcionários e professores do PESC pela estrutura fornecida para o desenvolvimento do trabalho.

Se segui em frente e cheguei aqui, foi porque houve quem acreditou em mim quando eu mesmo já não acreditava mais.

Mas uma pessoa merece um agradecimento especial. Essa pessoa que foi embora de repente, sem avisar e que, a cada dia, faz mais falta. A pessoa que sempre esteve do meu lado me apoiando, me ensinando que a vida é um tanto quanto qual complicada, e como é. Que sempre vinha assobiando com um sorriso no rosto, não importava a quantidade de problemas que nos cercavam.

Vô, muito obrigado por tudo. Esteja onde estiver, espero que esteja orgulhoso.

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

COMPARAÇÃO DE DESEMPENHO ENTRE OS MODELOS NEURAIS ÁGEIS ELM E WISARD

Luiz Fernando dos Reis de Oliveira

Março/2017

Orientador: Felipe Maia Galvão França

Programa: Engenharia de Sistemas e Computação

Modelos neurais são populares na área de aprendizado de máquina. Dentre os vários tipos de modelos desta classe, os modelos neurais ágeis se destacam por apresentarem tempo de treinamento consideravelmente inferior, sendo utilizados principalmente em domínios de aprendizado *online*. Dois exemplos deste tipo de modelo são a *Extreme Learning Machine* (ELM), que é uma rede neural com uma única camada oculta cujos pesos sinápticos não precisam ser ajustados, e a **Wi***lkes,* **S***tonham and* **A***leksander* **R***ecognition* **D***evice* (WiSARD), um modelo de rede neural sem pesos com múltiplos discriminadores que utilizam neurônios implementados como estruturas de memória RAM. Neste trabalho, é realizado um estudo comparativo entre os modelos neurais ágeis ELM e WiSARD, visando avaliar o desempenho de ambos quando aplicados a diferentes conjuntos de dados com diferentes características. A avaliação é feita a partir da comparação das métricas de acurácia de teste, tempos de treinamento e de teste, além do uso de memória RAM dos dois modelos.

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

PERFORMANCE COMPARISON BETWEEN THE AGILE NEURAL MODELS ELM AND WISARD

Luiz Fernando dos Reis de Oliveira

March/2017

Advisor: Felipe Maia Galvão França

Department: Systems Engineering and Computer Science

Neural models are popular in machine learning. Agile neural models are a subset of this kind of models and are characterized by presenting a significantly faster training time, being applied mainly in online learning domains. Two examples of agile neural models are the Extreme Learning Machine (ELM), a single hidden layer feedforward neural network which synaptic weights do not need to be iteractively adjusted, and the **Wi**lkes, **S**tonham and **A**leksander **R**ecognition **D**evice (WiSARD), a weightless neural network model with multiple discriminators that use neurons based on RAM memory structures. In this work, a comparative study between ELM and WiSARD models is made, aiming to evaluate both models performance when applied to different datasets having different characteristics. The evaluation is made by comparing test accuracy, training and testing times metrics, as well as the amount of RAM memory consumed by the models.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Redes neurais artificiais são modelos matemáticos com inspiração biológica na estrutura do cérebro humano [19] e constituem, computacionalmente, uma generalização do modelo *perceptron* [1]. Este tipo de modelo é popular na área de aprendizado de máquina em função de sua capacidade de aproximar qualquer função complexa [1], sendo a base de diversas técnicas como o *perceptron* de múltiplas camadas. No entanto, uma característica destes algoritmos é o tempo de treinamento que, em função da utilização de técnicas de minimização de erro que demoram para convergir, inviabiliza a utilização dos modelos em situações onde o aprendizado deva ocorrer em tempo real [29]. Para este tipo de domínio, é necessária a aplicação de modelos ágeis, dentre os quais estão a *extreme learning machine* (ELM) e a Wilkes, Stonhan *and* Aleksander *Recognition Device* (WiSARD).

1.1 Motivação

Há grande interesse em melhorias de técnicas mais recentes e sofisticadas relacionadas tanto a teoria do aprendizado quanto a aspectos de implementação. Um típico exemplo está em *deep learning*, que tem chamado muita atenção em função dos avanços em computação paralela para lidar com a redução do alto tempo de treinamento necessário para os modelos deste conjunto de técnicas [41]. Entretanto, o tempo de treinamento se mostra ineficiente para aplicações em que os dados de treinamento chegam em um fluxo e o aprendizado deve ser realizado em intervalos de tempo muito pequenos, o que motiva estudos sobre modelos com melhor performance de treinamento, como a ELM e a WiSARD. Estes modelos dividem propriedades interessantes como o baixo tempo de treinamento, a fácil implementação e a utilização de um mapeamento pseudoaleatório da entrada. Ao mesmo tempo, possuem particularidades como a possibilidade de implementação da WiSARD diretamente em *hardware* utilizando-se álgebra de Boole e o aprendizado em uma etapa da ELM. Os dois modelos apresentam como principal vantagem o baixo número de

parâmetros a serem configurados quando comparados a outros algoritmos da literatura, além da boa capacidade de generalização utilizando mapeamentos aleatórios em suas entradas.

Trabalhos recentes reforçam o estudo sobre a agilidade destes dois modelos. Em [7], aplica-se uma adaptação do modelo WiSARD ao problema da análise de crédito e compara-se seu desempenho com a o método *support vector machine* (SVM), uma técnica clássica que também possui forte popularidade e que busca criar um hiperplano de separação com uma margem máxima para separar o conjunto de dados em duas classes distintas [42]. Neste trabalho, os autores mostram que o modelo neural sem peso consegue acurácia equivalente, mas com tempo de treinamento duas ordens de magnitude menor. Além disso, possui maior flexibilidade por ter o mesmo desempenho em termos de acurácia em cenários de aprendizado online.

De maneira similar, em [28] compara-se o desempenho da ELM com os algoritmos SVM e *least squares support vector machine* (LS-SVM) [40], aplicando-os a diversos conjuntos de dados extraídos de repositórios públicos e voltados para problemas de regressão e classificação binária e multiclasse. O estudo mostrou que o modelo consegue atingir performance de classificação similar ou superior ao das duas técnicas, mas com tempos de treinamento aproximadamente três ordens de magnitude mais rápido.

1.2 Objetivos e Contribuições

O objetivo deste trabalho é realizar um estudo comparativo entre a WiSARD e a ELM, utilizando-os em um conjuntos de dados obtidos do repositório público UCI [34] e avaliando o desempenho dos modelos com relação à acurácia, tempo de treinamento, tempo de teste e uso de memória.

A principal contribuição do trabalho está em mostrar que o modelo WiSARD se apresenta de forma competitiva no cenário de modelos ágeis, comparando-o com o modelo de referência ELM. Com os resultados obtidos, é possível constatar que o modelo WiSARD consegue atingir resultados de acurácia equivalentes ao do modelo ELM com tempo de treinamento inferior. Contudo, a WiSARD não apresentou vantagens em tempo de teste em cenários com número elevado de classes.

1.3 Organização

Esta dissertação está organizada em cinco capítulos, dos quais este é o primeiro. No Capítulo 2, são descritos os principais conceitos teóricos envolvidos, apresentando os dois modelos mencionados. No Capítulo 3, apresenta-se a metodologia que guiou os experimentos, bem como as demais estratégias comparadas. No Capítulo 4, os

resultados obtidos são expostos, seguidos de discussões e análises sobre os mesmos. Finalmente, no Capítulo 5, são tiradas as devidas conclusões e possíveis trabalhos futuros são apontados. O trabalho conta ainda com o Apêndice A, que trata da exploração do espaço de configurações do modelo WiSARD, e o apêndice B, que apresenta o artigo aceito para publicação no 25º Simpósio Europeu de Redes Neurais Artificiais, Inteligência Computacional e Aprendizado de Máquina (ESANN 2017).

Capítulo 2

Modelos Neurais Ágeis

Neste capítulo são apresentados os modelos ELM e WiSARD. Para ambos os modelos, inicialmente apresenta-se os conceitos teóricos gerais, seguindo-se de uma explicação mais aprofundada sobre os mesmos. Por fim, fala-se sobre trabalhos relacionados e discute-se a respeito de similaridades entre os dois modelos.

2.1 *Extreme Learning Machine*

Dentre os vários tipos de redes neurais artificiais, as redes do tipo *feedforward* são as que apresentam maior popularidade e cujo nome se dá em função das ligações entre os neurônios não formarem um ciclo. A arquitetura deste tipo de rede baseia-se em (i) uma camada de neurônios de entrada; (ii) uma ou mais camadas denominadas **ocultas** e (iii) uma camada de saída.

Uma das características mais atrativas deste modelo é a capacidade de modelar qualquer função complexa, embora apresentem um alto custo computacional da otimização dos pesos sinápticos durante a etapa de treinamento. Algoritmos de aprendizado como retropropagação tendem a apresentar tempo de treinamento elevado.

2.1.1 O modelo ELM

O objetivo da *extreme learning machine* é promover um modelo de aprendizado unificado para diferentes tipos de arquiteturas de redes [24]. Em sua essência, a ELM é caracterizada por não exigir que os pesos sinápticos dos neurônios de sua camada oculta sejam otimizados.

Dada uma SLFN com L neurônios, a função de ativação da rede é dada por $\sum_{i=1}^{L} \beta_i g_i(x) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x)$, em que $g_i(x)$ representa a função de ativação do i-ésimo neurônio. A ELM pode utilizar vários tipos de função de ativação, como por exemplo funções aditivas, funções de base radial e até mesmo funções de *kernel*. [28]

Dado um conjunto de dados com N observações da forma $(x_i, t_i) \in R^d \times R^m$, e baseando-se no teorema da interpolação , uma SLFN com L neurônios é capaz de aproximar essas N observações com erro zero. Essa aproximação é descrita pela seguinte equação:

$$\sum_{i=1}^{L} \beta_i G(a_i, b_i, x_j) = t_j, \text{ j = 1, ..., N}$$

A fórmula pode ser reescrita como $H\beta = T$, de forma que os termos referentes à função de ativação sejam agrupados na forma da matriz H. Esta matriz é, geralmente, referenciada como **matriz de mapeamento aleatório de atributos**, e é dada por:

$$H = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_L, b_L, x_1) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_N) & \cdots & G(a_L, b_L, x_N) \end{bmatrix}$$

Os componentes de H são definidos da seguinte forma: $a_i \in R^d$ e $b_i \in R^+$ são parâmetros gerados aleatoriamente. x_i é uma observação do conjunto de dados $X = (x_i, t_i)|x_i \in R^d, t_i \in R^m, i = 1, ..., N$, onde m é o número de atributos envolvidos, e $G(a_i, b_i, x)$ é a função de saída i-ésimo nó oculto. Nesta matriz, a i-ésima coluna representa o i-ésimo neurônio de saída com respeito às observações do conjunto de entrada, enquanto a j-ésima linha representa o mapeamento aleatório de características com relação à entrada j.

Visto que os parâmetros (a_i, b_i) são gerados de forma aleatória e permanecem fixos e que o conjunto de dados também é fixo, resta apenas descobrir o valor de β. Este valor pode ser obtido através do cálculo do estimador por mínimos quadrados:

$$H\beta = T \rightarrow ||H\hat{\beta} - T|| = min_\beta ||H\beta - T||$$

Se $L = N$, a matriz H possui inversa e a rede pode aproximar os dados com erro zero. No entanto, esta igualdade geralmente não ocorre. Desta forma, a solução da equação pode ser dada por:

$$\beta = H^\dagger T$$

$H^\dagger$ é a **inversa generalizada de Moore-Penrose**. Vários métodos podem ser utilizados para o cálculo dessa matriz como o método da projeção ortogonal, o método da ortogonalização, decomposição em valores singulares (SVD) [5]. Em geral, o método da projeção ortogonal é preferível [28], sendo definido como $H^\dagger = (H^T H)^{-1} H^T$ se $H^T H$ é não-singular ou $H^\dagger = H^T (H H^T)^{-1}$ se $H H^T$ é não-singular.

A principal característica que envolve o funcionamento do modelo ELM está na transformação não linear no conjunto de entrada através de um mapeamento

aleatório. Esta transformação causa uma elevação considerável na dimensionalidade dos dados, possibilitando traçar um hiperplano que separe de maneira mais eficiente as classes do conjunto. No entanto, como descrito em [1], o preço pago pelo aumento da capacidade de generalização dentro da amostra (E_{in}) é a redução da capacidade de generalização fora dela (E_{out}).

2.1.2 Fator de Regularização

Segundo analisado em [36], sugere-se ainda a adição de um valor positivo $\frac{1}{C}$ à diagonal principal da matriz $H^T H$ (ou HH^T). Este valor é um fator de regularização, que contribui para a melhoria da performance de generalização. Esta adição implica em alterar a equação de forma que $\beta = (\frac{I}{C} H^T H)^{-1} H^T T$ se $H^T H$ é não-singular ou $\beta = H^T (\frac{I}{C} HH^T)^{-1} seT\ HH^T$ é não-singular.

Nestas equações, I é a matriz identidade de ordem L. No trabalho citado, os autores se baseiam na teoria de decaimento de pesos [1] para propor um termo de regularização para o modelo ELM original, conseguindo obter melhores resultados em um conjunto de *benchmark* de problemas de regressão.

2.1.3 Algoritmo

Conforme descrito em [18], o algoritmo básico da ELM - ilustrado pelo Algoritmo 1 - consiste em: (i) gerar aleatoriamente os parâmetros (a_i, b_i) dos neurônios da camada oculta, (ii) criar a matriz oculta de saída **H** com base no conjunto de dados de entrada, e (iii) calcular a matriz de pesos de saída β.

Dados: Conjunto de treinamento, função de saída G e número de neurônios L
Resultado: Matriz de pesos β
início
  Gerar aleatoriamente parâmetros dos nós ocultos (a_i, b_i), i = 1, ..., L
  Calcular a matriz de saída da camada oculta **H**
  Calcular a matriz de pesos de saída β tal que:
$$\beta = H^{\dagger} T$$
fim

Algoritmo 1: ELM básico

Para realizar a classificação, é necessário gerar uma nova matriz de mapeamento H_{teste} baseada no conjunto de teste $X = (x_i, y_i)$, i = 1, ..., N. A performance do modelo é obtida calculando-se o erro entre a saída $T_o = H^{\dagger}_{teste} \beta$ com a saída real Y. Visto que o modelo ELM pode ser utilizada tanto para problemas de regressão

quanto para problemas de classificação, o erro é calculado através do erro médio quadrático para problemas da primeira classe, enquanto para a segunda utiliza-se a contagem de classificações incorretas.

2.1.4 Trabalhos Relacionados

Grande parte dos trabalhos relacionados à ELM são referentes ao desenvolvimento teórico do modelo, além de evoluções e adaptações do mesmo. Em [31] o autor explora a combinação do modelo ELM com algoritmos de redução de dimensionalidade em um framework que possibilite a execução da tarefa em tempo reduzido. Evoluções do modelo são tratadas em [27], com o desenvolvimento de uma versão incremental do algoritmo que pode ser utilizada em aprendizado online, enquanto [33], trata da aplicação do modelo em um domínio de valores complexos.

No entanto, embora em quantidade menor, alguns trabalhos tratam da aplicação do modelo a problemas práticos. Em [12], os autores desenvolvem uma aplicação de reconhecimento facial utilizando o modelo ELM. Já em [44], o modelo é utilizado no motor de classificação da aplicação de reconhecimento de placas de trânsito.

Trabalhos recentes iniciam estudos da aplicação da ELM em ambientes de hardware. Em [11], os autores utilizam a ELM em uma *brain-machine*, obtendo bons resultados de classificação comparados a algoritmos da literatura. Já em [43] e [39] os autores exploram a utilização de modelos arquiteturais com consumo energético mais eficiente.

O principal autor do modelo disponibiliza uma lista de temas ainda em aberto relacionados à ELM. Alguns estão relacionados à provas teóricas como a sensibilidade do algoritmo ao número de neurônios e ao estudo da fronteira de generalização. Outros temas envolvem conceitos mais técnicos como escalabilidade e implementação do algoritmo em ambientes de paralelismo e distribuição.

2.2 WiSARD

O trabalho desenvolvido em [6] apresenta, em 1959, um dispositivo capaz de realizar o reconhecimento de caracteres e dígitos. Este dispositivo é composto por um mosaico fotocelular de tamanho (10×15) que decompões a entrada em grupos de células e registra os subpadrões aprendidos.

Baseado neste trabalho, Wilkes, Stonhan e Aleksander apresentam, em 1984, o primeiro dispositivo comercial para reconhecimento de padrões: o modelo WiSARD (acrônimo para **W**ilkes, **S**tonhan and **A**leksander's **r**ecognition **d**evice) [3]. Diferentemente dos modelos neurais apresentados na seção anterior, o modelo WiSARD pertence à categoria das redes neurais sem peso (RNSP).

O modelo é composto por um conjunto de discriminadores, onde cada discriminador é responsável pelo reconhecimento de uma classe específica. Cada discriminador possui um conjunto de neurônios que possuem a estrutura de uma memória RAM, isto é, possuem endereços que armazenam valores. Além disso, cada discriminador possui um mapeamento pseudoaleatório, que é utilizado para mapear os elementos do padrão de entrada para os neurônios. O modelo WiSARD opera iterativamente, isto é, ele processa cada observação isoladamente. Este aspecto do modelo faz com que o mesmo possua etapas de treinamento e classificação bem definidas.

2.2.1 Etapa de Treinamento

Nesta etapa, deseja-se que o modelo aprenda a reconhecer um determinado padrão relacionado a uma classe específica. O modelo WiSARD foi projetado para reconhecer padrões binários. Desta forma, é necessário realizar um pré-processamento da entrada com o objetivo de transformá-la em um conjunto de bits.

Como a entrada pode conter tanto atributos categóricos quanto contínuos, há diferentes maneiras de transformá-la em uma sequência binária. Para os atributos categóricos, uma alternativa é, definido o valor k referente ao número de bits do atributo, criar tantos grupos de k bits quantas forem os possíveis valores do atributo, atribuindo o valor 0 em todos os bits dos conjuntos diferentes do valor original do atributo e 1 em todos os bits do conjunto igual ao valor original.

Para atributos contínuos, pode-se utilizar o termômetro, onde, definido o valor k, cria-se um conjunto de k intervalos e normaliza-se o valor original do atributo dentro de uma escala definida pelos menor e maior valores possíveis daquele atributo. Todos os intervalos que agrupem valores menores ou iguais ao valor original do atributo recebem o valor 1, enquanto os intervalos maiores recebem o valor 0.

A partir da apresentação do padrão binário, realiza-se um mapeamento pseudoaleatório dos bits deste padrão de entrada. Cada discriminador possui seu próprio mapeamento e, após criado, não se altera. Dada a nova ordem dos bits de entrada, realiza-se a fragmentação do padrão em n partes - denominadas tuplas - e cada parte é relacionada para uma estrutura RAM.

Inicialmente, todos os endereços das estruturas RAM estão zeradas. Dado o valor da tupla apresentado, o endereço correspondente passa a conter o valor 1, indicando que aquele padrão foi visto pela rede.

A Figura 2.1 mostra as etapas do treinamento dos neurônios de um discriminador dado um padrão de entrada. O exemplo em questão é o aprendizado de figuras de dígitos, onde um discriminador está aprendendo o padrão do dígito 1.

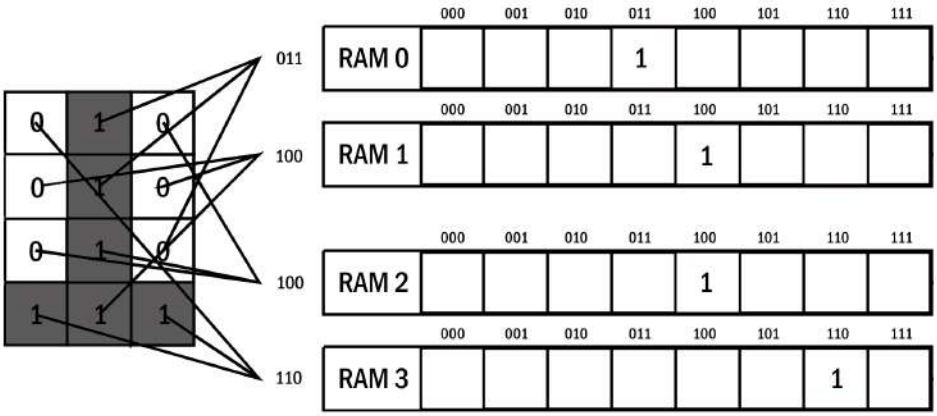

Figura 2.1: Treinando um discriminador da WiSARD

2.2.2 Etapa de Classificação

Nesta etapa, há uma observação que deseja ser classificada. Ela deve ser apresentada a cada um dos discriminadores que forma a WiSARD, de forma que cada um retorne um valor de resposta conhecido como **grau de ativação**. A entrada será decomposta para formação das tuplas seguindo o mapeamento pseudo-aleatório de cada discriminador, de forma que seja verificado se existe um valor diferente de 0 dado o endereço da tupla. Se houver, a RAM é ativada e uma unidade é acrescida ao grau de ativação.

Dado o conjunto de respostas, o padrão escolhido é aquele que possuir o maior grau de ativação, denotado por T_{max}. Além disso, é possível calcular a confiança γ da resposta do discriminador, dada por $\gamma = \frac{T_{max} - T_{max-1}}{T_{max}}$. A Figura 2.2 ilustra o processo de classificação do padrão apresentado.

2.2.3 *Bleaching*

Um problema enfrentado pelo modelo WiSARD é conhecido como **saturação**, que é quando, após uma quantidade excessiva de treinamento, muitos dos possíveis endereços das RAM dos discriminadores estão preenchidas. Isto faz com que mais de um discriminador responda aos estímulos e que leva ao modelo ficar em dúvida com relação a qual classe o padrão apresentado pertence.

Para contornar este problema foi desenvolvida a técnica de *bleaching*, que consiste em alterar o funcionamento dos endereços. Ao invés de registrar o valor 1, o endereço passa a operar como um contador: uma vez que uma sequência de bits seja apresentada a um endereço, o valor anterior é incrementado em 1.

Com isso, deve-se definir um **limiar** de forma que, dada uma tupla, apenas as estruturas RAM cujo valor armazenado no dado endereço seja maior ou igual ao limiar são ativadas. A Figura 2.3 ilustra o funcionamento da técnica. Variações da

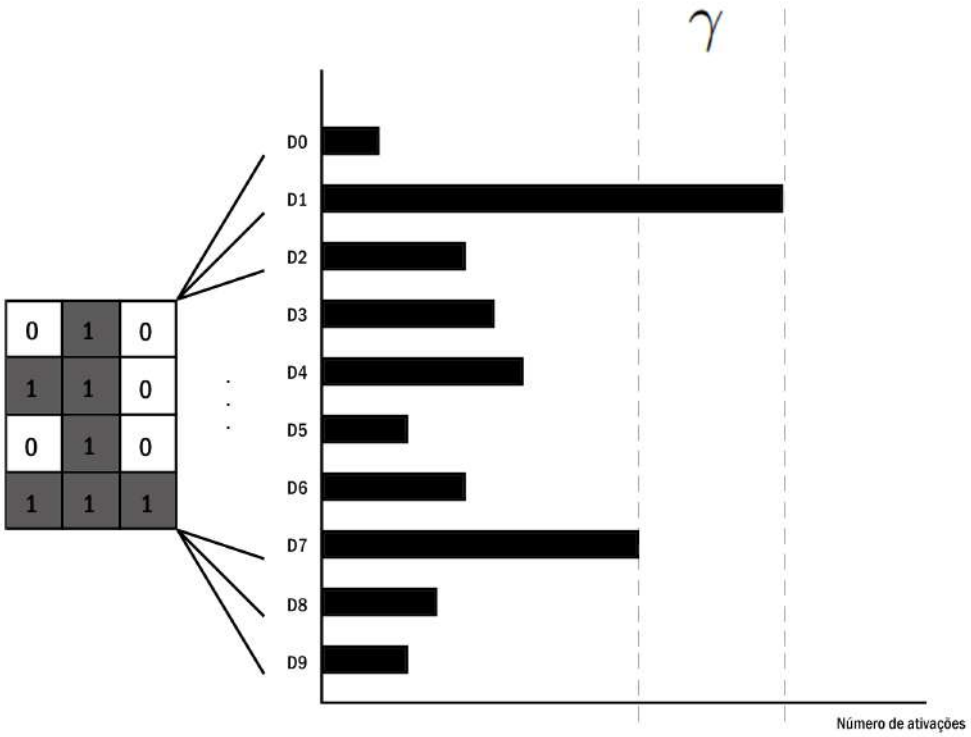

Figura 2.2: Classificando um padrão utilizando WiSARD.

técnica de *bleaching* foram exploradas em [10].

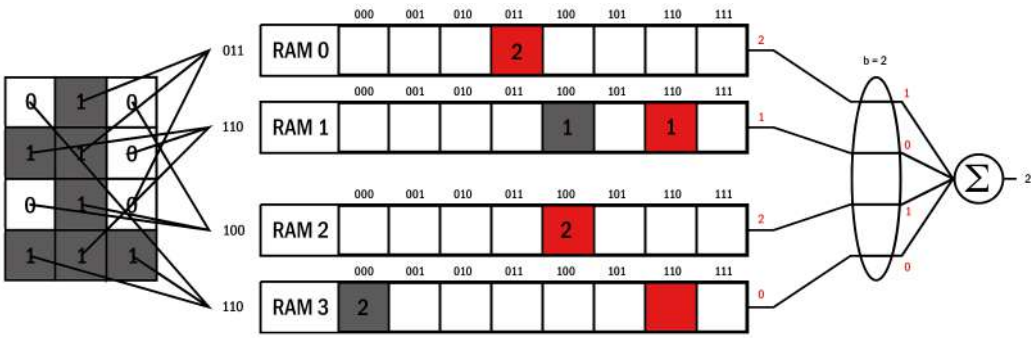

Figura 2.3: Classificando um padrão utilizando bleaching.

2.2.4 Implementação

Uma das características do modelo WiSARD é a facilidade de implementação do mesmo, tendo em vista seu desenvolvimento voltado para aplicações em *hardware*. A implementação em *software* do modelo pode ser feita seguindo o diagrama de

classes apresentado na Figura 2.4.

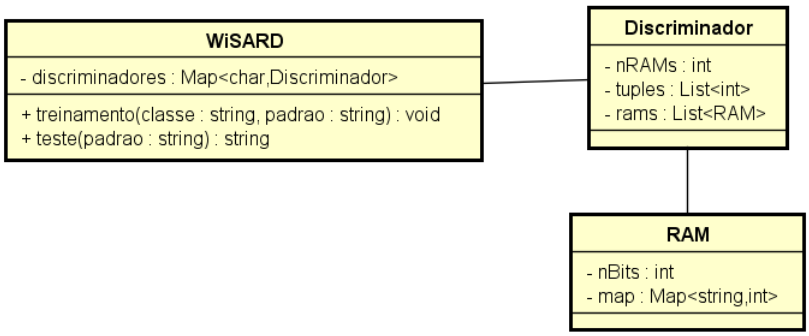

Figura 2.4: Um diagrama de classes do modelo WiSARD.

Uma questão a ser observada é a forma de implementação das estruturas de memória RAM. Dado que as tuplas possuem tamanho **n** cada uma das estruturas RAM possuirá, consequentemente, 2^n endereços de memória. A depender da combinação do quão grande o valor **n** seja, em conjunto com o número de estruturas RAM, a implementação do modelo pode acabar por consumir todos os recursos da memória física do computador. Além disso, em certas ocasiões, é provável que uma quantidade muito grande dos endereços não contenha um acesso sequer.

Para contornar este problema, a proposta é a utilização de um **mapa** - que é uma estrutura de par **(chave, valor)** - para representar a RAM, visto que é uma estrutura que se comporta de forma semelhante ao conceito original da RAM, mas que cresce dinamicamente. Nesta estrutura, a chave é representada pelo endereço de memória a ser acessado, enquanto o valor é a quantidade de acessos a este endereço.

Quando um acesso a algum endereço é feito, verifica-se no mapa se o endereço já existe. Se não existir, significa que um novo endereço está sendo acessado, sendo necessária sua criação. Este procedimento é feito atribuindo ao mapa o par **(endereço, 0)**. Caso o endereço exista, é necessário apenas incrementar o valor da chave em 1.

2.2.5 Trabalhos Relacionados

Diversos trabalhos utilizam o modelo WiSARD ou alguma variação do modelo como classificador em aplicações. No contexto de aprendizado *online*, em [16] utiliza-se o modelo associado a memórias com janela de tempo de aplicado ao problema de rastreamento de objetos, enquanto em [15] utiliza-se o modelo para rastreamento de músicas. No domínio de prognóstico médico, em [14] utiliza-se a WiSARD para

detecção de crises epiléticas.

Uma variação da WiSARD para abordar problemas de clusterização, denominada clus-WiSARD, é utilizada em [7] para o domínio da análise de crédito, comparando o modelo com a técnica SVM e mostrando diversas vantagens do modelo sem pesos. A mesma variação é utilizada em [8] clusterização de fluxo de dados, reforçando o bom desempenho do modelo em aprendizado *online*.

Outra variação da WiSARD, denominada DRASiW, também é utilizada em vários trabalhos. Este modelo é capaz de explicitar o conhecimento da rede e gerar a chamada **imagem mental** [21]. Em [13], a variação foi utilizada na extração de regras em bases de dados. Já no contexto da teoria do aprendizado, um estudo inicial realizado em [20] tenta realizar a clonagem do conhecimento de um modelo DRASiW utilizando uma técnica denominada *memory transfer*. Neste trabalho, a transferência de conhecimento obtém pouca perda da capacidade de generalização, sendo uma primeira exploração deste tipo de modelo aplicados a *transfer learning*.

2.3 Características Comuns aos Modelos

Embora sejam baseados em estruturas diferentes - a SLFN para a ELM e estruturas de memória RAM para a WiSARD - os dois modelos dividem características comuns e que os tornam muito atrativos, como por exemplo sua facilidade de implementação e o número reduzido de parâmetros a serem otimizados. Além disso, por evitarem métodos de otimização em seus algoritmos, o tempo de treinamento de ambos os modelos é consideravelmente baixo.

No entanto, ainda há particularidades que diferem os modelos, como a possibilidade de aplicação da ELM a problemas de regressão, assunto ainda em aberto com relação a redes sem peso. Ainda, apesar de estudos recentes que aplicam o modelo ELM em arquiteturas de hardware específico, o modelo WiSARD foi desenvolvido tendo como base um projeto de hardware, de forma que pode ser implantado de forma simples em diversos cenários.

Os modelos também são caracterizados pela utilização de mapeamentos pseudo-aleatórios, estando assim ligadas ao chamado *Reservoir computing* [38], que é um modelo de computação onde os sinais de entrada são conectados de forma aleatória. Modelos deste gênero - cujos representantes mais populares são a *Liquid-state machine* [35] e a *Echo state network* [30] - também apresentam como característica o baixo tempo de treinamento, sendo comumente utilizadas em aplicações de aprendizado em tempo real.

Capítulo 3

Metodologia Experimental

Neste capítulo, é feita a descrição de toda a metodologia utilizada na comparação dos modelos. Inicialmente, uma breve descrição dos conjuntos de dados é realizada, seguindo-se de uma explicação mais aprofundada sobre os métodos utilizados na análise. Por fim, descreve-se o ambiente computacional utilizado para realização dos experimentos.

3.1 Conjuntos de Dados

Para comparar o desempenho dos modelos, foram utilizados 14 conjuntos de dados provenientes do repositório público UCI [34]. Estes conjuntos correspondem a uma parcela do total de conjuntos utilizados em [28] e possuem características variadas com relação ao número de atributos, número de classes, tamanho do conjunto e tipos dos atributos. Para fins de análise, os conjuntos de dados são divididos quanto ao número de classes: binários e multiclasse. A Tabela 3.1 apresenta um resumo das principais características dos conjuntos descritos.

3.1.1 Binários

Adult

Consite em classificar se uma pessoa ganha mais de US$50 mil por ano ou não. A base formada por 14 atributos, sendo seis contínuos e oito categóricos. Possui 48842 observações, com 32561 destas voltadas para treinamento. A base possui atributos faltantes, que foram retirados dos experimentos.

Australian Credit

Consiste em classificar se uma pessoa terá ou não seu pedido de crédito aprovado. Possui 690 observações com 14 atributos, sendo 6 numéricos e 8 categóricos.

Banana

Base de dados que contém 5300 observações de *clusters* em formatos de banana. Possui 2 atributos contínuos que correspondem aos eixos x e y, respectivamente, do formato do cluster. Não há atributos faltantes.

Diabetes

Classifica se uma determinada pessoa apresenta ou não quadro de diabetes de acordo com parâmetros estabelecidos pela Organização Mundial da Saúde (OMS). Contém 768 observações, possui 8 atributos numéricos e há atributos faltantes.

Liver

Apresenta registros de exames de sangue de indivíduos do sexo masculino e o número de doses de bebidas alcoólicas ingeridas pelo mesmo para classificar possíveis desordens no fígado. Cada uma das 45 observações possui 6 atributos. Não há atributos faltantes.

Mushroom

Consiste na classificação de observações hipotéticas de cogumelos dos gêneros Agaricus e Lepiota entre as classes **comestível** e **venenoso**. Possui um total de 22 atributos, todos nominais. Há atributos faltantes, todos relacionados ao 11º atributo. Há um total de 8124 observações.

3.1.2 Multiclasse

Ecoli

Base com 336 observações para predição de sítios de localização de proteínas em bactérias. Possui 7 atributos, nenhum faltante. As observações estão divididas em 7 classes, sendo que a maior parte da base - 143 - está concentrada em uma classe.

Glass

Classificação do tipo de vidro, com motivação para uso em investigações de crimes. Possui 124 observações com 9 atributos e divididas entre 7 classes.

Iris

Consiste em verificar a qual espécie de flor de Iris a observação pertence, dentre três possíveis. Possui quatro atributos contínuos e 150 observações.

Letter

Base com 20.000 observações de imagens de letras maiúsculas, divididas entre as 26 letras do alfabeto, criadas com base em 20 tipos de fontes diferentes e com algumas distorções. As observações possuem 16 atributos, que são estatísticas retiradas das imagens originais.

Satimage

Base de dados com observações de valores multi-espectrais em imagens de satélite, cujo objetivo é a classificação associada ao pixel central em uma vizinhança de pixels 3x3. Este *pixel* está associado ao tipo de solo que a imagem representa dentre 6 possíveis. Possui 4.435 observações de treino e 2.000 de teste, cada uma com 36 atributos numéricos no intervalo (0,255). Não há dados faltantes.

Segment

Possui 2.310 observações de segmentos de imagens de *outdoors* com 19 atributos contínuos, onde o objetivo é classificar a qual dos 7 *outdoors* a imagem pertence.

Vehicle

Consiste na classificação de silhuetas de veículos, dentre 4 possíveis. Possui 846 observações com 18 atributos cada.

Wine

Base para reconhecimento de três tipos de vinho. Possui 13 atributos contínuos e não há dados faltantes dentre as 178 observações.

3.2 Metodologia

Alguns conjuntos de dados estão divididos entre conjunto de treino e teste, enquanto outros são compostos por um único grande conjunto. Para efetuar a comparação do desempenho dos dois modelos neste último tipo, optou-se por utilizar duas metodologias: a primeira utiliza validação cruzada dividindo-se o conjunto em 10 partes. A segunda, utilizada em [28], consiste em realizar uma permutação aleatória entre as observações do conjunto e dividir o conjunto em 2/3 para treinamento e 1/3 para teste. Os resultados apresentados são dados pela média aritmética de 20 execuções independentes. Para utilização no modelo ELM, os dados foram normalizados no intervalo [-1, 1].

Conjunto	Tamanho	Atributos	Classes
Adult	48842	14	2
Australian Credit	690	14	2
Banana	5300	2	2
Diabetes	768	8	2
Liver	45	6	2
Mushroom	8124	22	2
Ecoli	336	7	7
Glass	124	9	7
Iris	150	4	3
Letter	20000	16	26
Satimage	6435	36	6
Segment	2310	19	7
Vehicle	846	18	4
Wine	178	13	3

Tabela 3.1: Resumo das características dos conjuntos de dados

Para medir a sensibilidade dos modelos quanto ao número de atributos variados, não foi realizado qualquer pré-processamento no sentido de reduzir a dimensionalidade dos dados. Além disso, há conjuntos com atributos faltantes em algumas observações. Neste caso, optou-se pelo descarte destas observações.

É válido ressaltar que a utilização de duas metodologias de avaliação visa tornar a comparação mais justa uma vez que, para a segunda metodologia, parâmetros ótimos já foram descobertos para o modelo ELM, enquanto este tipo de estudo ainda se faz necessário para o modelo WiSARD. A Tabela 3.2 apresenta as configurações utilizadas. A mesma configuração foi utilizada nas duas metodologias visando mostrar o impacto da falta de otimização de parâmetros.

Conjunto	C
Adult	2^{-6}
Australian Credit	2^{-1}
Banana	2^{22}
Diabetes	2^{-2}
Liver	2^{1}
Mushroom	2^{10}
Ecoli	2^{2}
Glass	2^{3}
Iris	2^{5}
Letter	2^{10}
Segment	2^{10}
Satimage	2^{7}
Vehicle	2^{7}
Wine	2^{0}

Tabela 3.2: Valores escolhidos para o fator de regularização C

3.3 Ambiente Computacional

Para executar os experimentos, foi utilizado um computador contendo um processador Intel Core i7-2600K, 3.40GHz, 8GB de memória RAM e sistema operacional Ubuntu 14.04. Foram desenvolvidos programas utilizando a linguagem Python 2.7. O código da ELM foi baseado no código original escrito em Matlab e disponibilizado em `http://www.ntu.edu.sg/home/egbhuang/elm_codes.html`. A análise de memória foi feita utilizando-se a ferramenta Pympler, cuja documentação pode ser encontrada em `http://pythonhosted.org/Pympler/`.

Capítulo 4

Resultados e Discussão

Neste capítulo são apresentados os resultados computacionais obtidos a partir da aplicação dos modelos WiSARD e ELM aos conjuntos de dados descritos no capítulo anterior. Cada uma das métricas estudadas - acurácia, tempos de treinamento e teste e uso de memória são discutidos individualmente. Em todas as tabelas, destaca-se o melhor valor em negrito.

4.1 Acurácia

As acurácias de teste dos modelos seguindo a estratégia de divisão 2/3 para treino e 1/3 para teste podem ser verificados na Tabela 4.1 para os conjuntos de dados binários e na Tabela 4.2. A ELM obteve melhor desempenho em todos os conjuntos de dados explorados, embora a WiSARD tenha conseguido acompanhar a performance na maioria das bases de dados com uma diferença em média de 4%.

		ELM		**WiSARD**	
Conjunto	**Tamanho**	**Acurácia**	**DesvPad**	**Acurácia**	**DesvPad**
Adult	48842/32561	**0.802**	0.001	0.773	0.017
Australian	690	**0.739**	0.027	0.687	0.037
Banana	400/4900	**0.877**	0.001	0.848	0.014
Diabetes	768	**0.765**	0.025	0.706	0.026
Liver	45	**0.722**	0.033	0.570	0.054
Mushroom	8124	**0.859**	0.002	0.850	0.007

Tabela 4.1: Acurácia de teste dos conjuntos de dados binários; melhores valores em negrito. Resultados obtidos a partir da média de 20 execuções independentes. Conjuntos com apenas um valor na coluna **tamanho** sofreram permutações aleatórias das observações e foram utilizados 2/3 das mesmas para treinamento e 1/3 para teste. Conjuntos entre linhas mais grossas apresentam equivalência estatística.

Apresenta-se na Figura 4.1 e na Figura 4.2 gráficos que permitem melhor visualização das informações de acurácia para os conjuntos de dados binários e multiclasse,

		ELM		WiSARD	
Conjunto	**Tamanho**	**Acurácia**	**DesvPad**	**Acurácia**	**DesvPad**
Ecoli	336	**0.876**	0.025	0.811	0.027
Glass	124	**0.899**	0.039	0.875	0.033
Iris	150	**0.974**	0.019	0.935	0.046
Letter	20000	**0.932**	0.003	0.808	0.009
Satimage	4435/2000	**0.882**	0.005	0.848	0.008
Segment	2310	**0.962**	0.006	0.933	0.015
Vehicle	846	**0.828**	0.020	0.674	0.021
Wine	178	**0.980**	0.020	0.952	0.030

Tabela 4.2: Acurácia de teste dos conjuntos de dados multiclasse; melhores valores em negrito. Resultados obtidos a partir da média de 20 execuções independentes. Conjuntos com apenas um valor na coluna **tamanho** sofreram permutações aleatórias das observações e foram utilizados 2/3 das mesmas para treinamento e 1/3 para teste. Conjuntos entre linhas mais grossas apresentam equivalência estatística.

respectivamente. Nelas, é possível notar que, apesar da superioridade, existe equivalência estatística entre os modelos em alguns conjuntos de dados, como *Australian*, *Mushroom*, *Glass*, *Iris* e *Wine* quando se utiliza esta estratégia de divisão.

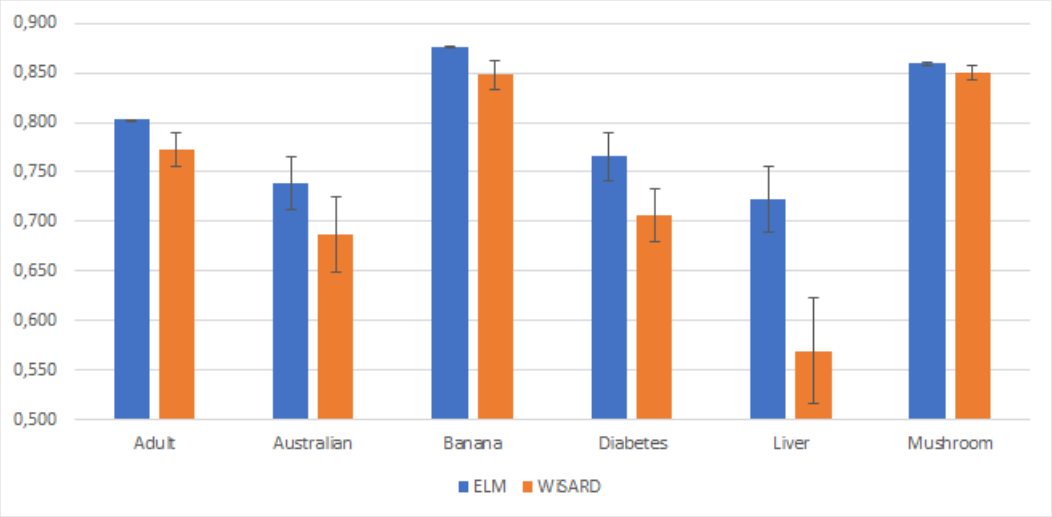

Figura 4.1: Comparação de acurácia média entre os modelos aplicados aos conjuntos de dados binários utilizando 2/3 dos dados para treinamento e 1/3 para teste, quando possível.

Um ponto a ser notado é o fato de que os desvios padrões da WiSARD se encontram, em geral, maiores que os da ELM. Experimentos iniciais mostram que este comportamento também está relacionado à configuração de *bits* do modelo. Analisando o conjunto *Banana*, o mesmo apresenta desvio padrão 0.014 utilizando 20 *bits* para cada atributo e tamanho de tuplas 20. Se os valores são alterados para 10, o desvio padrão apresentado passa a ser de 0.058, enquanto a alteração para 30

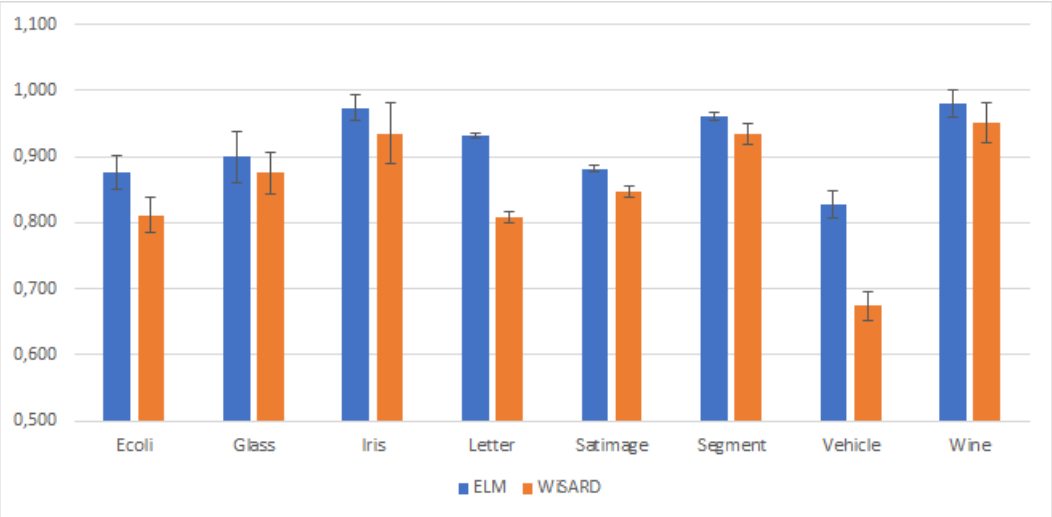

Figura 4.2: Comparação de acurácia média entre os modelos aplicados aos conjuntos de dados multiclasse utilizando 2/3 dos dados para treinamento e 1/3 para teste, quando possível.

apresenta o valor 0.027.

Já as acurácias de teste dos modelos seguindo a estratégia de validação cruzada podem ser verificados na Tabela 4.3 para os conjuntos de dados binários e na Tabela 4.4, que apresentam o tamanho dos conjuntos, a acurácia média e o desvio padrão. Nestes casos, é possível notar a variação entre qual modelo apresenta maior acurácia, o que ressalta a importância da otimização de configurações.

		ELM		**WiSARD**	
Conjunto	**Tamanho**	**Acurácia**	**DesvPad**	**Acurácia**	**DesvPad**
Adult	48842/32561	0.760	0.002	**0.778**	0.096
Australian	690	0.657	0.034	**0.661**	0.018
Banana	400/4900	0.794	0.003	**0.852**	0.052
Diabetes	768	0.699	0.056	**0.711**	0.034
Liver	45	0.496	0.018	**0.557**	0.036
Mushroom	8124	**0.859**	0.036	0.842	0.009

Tabela 4.3: Acurácia de teste dos conjuntos de dados binários. Melhores valores em negrito. Resultados obtidos a partir da média de 20 execuções independentes. Utilizou-se validação cruzada com 10 subconjuntos para conjuntos com apenas um valor na coluna **tamanho**. Conjuntos entre linhas mais grossas apresentam equivalência estatística.

Como anteriormente, ilustra-se os resultados dos conjuntos de dados binários na Figura 4.3 e para os conjuntos de dados multiclasse na Figura 4.4. Utilizando esta metodologia, é possível notar equivalência estatística entre os conjuntos de dados *Adult*, *Australian*, *Diabetes*, *Mushroom*, *Iris*, *Satimage*, *Vehicle* e *Wine*.

		ELM		WiSARD	
Conjunto	**Tamanho**	**Acurácia**	**DesvPad**	**Acurácia**	**DesvPad**
Ecoli	336	**0.839**	0.043	0.777	0.003
Glass	124	0.847	0.036	**0.889**	0.005
Iris	150	**0.960**	0.043	0.900	0.063
Letter	20000	**0.933**	0.022	0.801	0.054
Satimage	4435/2000	**0.899**	0.015	0.850	0.080
Segment	2310	0.699	0.068	**0.943**	0.006
Vehicle	846	**0.755**	0.036	0.691	0.056
Wine	178	0.850	0.033	**0.967**	0.054

Tabela 4.4: Acurácia de teste dos conjuntos de dados multiclasse. Melhores valores em negrito. Resultados obtidos a partir da média de 20 execuções independentes. Utilizou-se validação cruzada com 10 subconjuntos para conjuntos com apenas um valor na coluna **tamanho**. Conjuntos entre linhas mais grossas apresentam equivalência estatística.

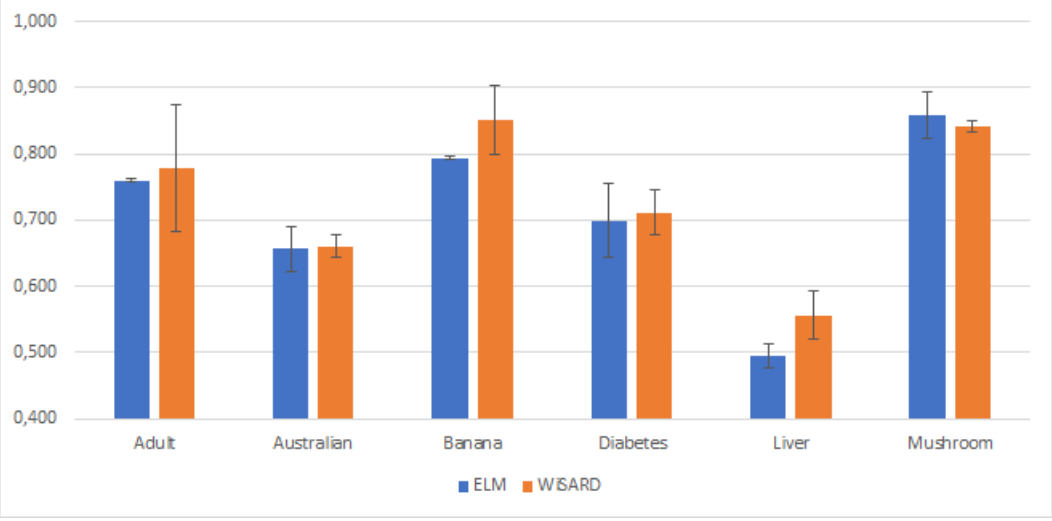

Figura 4.3: Comparação de acurácia média entre os modelos aplicados aos conjuntos de dados binários utilizando validação cruzada com 10 subconjuntos, quando possível.

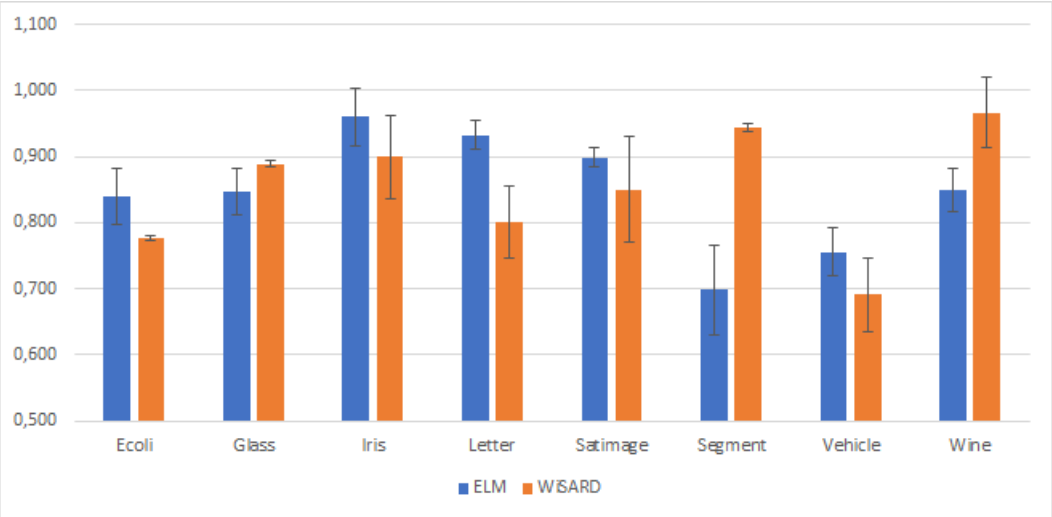

Figura 4.4: Comparação de acurácia média entre os modelos aplicados aos conjuntos de dados multiclasse utilizando validação cruzada com 10 subconjuntos, quando possível.

4.2 Tempo de Treinamento

Os tempos de treinamento dos modelos podem ser verificados na Tabela 4.5 para os conjuntos de dados binários e na Tabela 4.6, que apresentam o tamanho dos conjuntos, o tempo de treinamento - em segundos - e o desvio padrão. A WiSARD obteve performance de treino superior em todos os conjuntos de dados, chegando a reduzir em uma ordem de magnitude em diversos alguns casos.

		ELM		**WiSARD**	
Conjunto	**Tamanho**	**Treino**	**DesvPad**	**Treino**	**DesvPad**
Adult	48842/32561	8.160	0.873	**0.329**	0.002
Australian	690	0.817	0.110	**0.246**	0.028
Banana	400/4900	0.682	0.054	**0.029**	0.008
Diabetes	768	0.888	0.120	**0.366**	0.015
Liver	45	0.849	0.110	**0.125**	0.004
Mushroom	8124	1.379	0.120	**0.298**	0.002

Tabela 4.5: Tempos de treinamento dos conjuntos de dados binários (em segundos). Resultados obtidos a partir da média de 20 execuções independentes. Conjuntos com apenas um valor na coluna **tamanho** sofreram permutações aleatórias das observações e foram utilizados 2/3 das mesmas para treinamento e 1/3 para teste.

4.3 Tempo de Teste

Os tempos de teste dos modelos podem ser verificados na Tabela 4.7 para os conjuntos de dados binários e na Tabela 4.8, que apresentam o tomanho dos conjuntos,

		ELM		WiSARD	
Conjunto	**Tamanho**	**Treino**	**DesvPad**	**Treino**	**DesvPad**
Ecoli	336	0.781	0.072	**0.143**	0.006
Glass	124	0.717	0.148	**0.116**	0.002
Iris	150	0.618	0.048	**0.038**	0.001
Letter	20000	6.205	0.608	**1.339**	0.008
Satimage	4435/2000	2.567	0.199	**0.950**	0.013
Segment	2310	1.346	0.152	**0.284**	0.118
Vehicle	846	0.792	0.068	**0.634**	0.158
Wine	178	0.791	0.636	**0.128**	0.021

Tabela 4.6: Tempos de treinamento dos conjuntos de dados multiclasse (em segundos). Resultados obtidos a partir da média de 20 execuções independentes. Conjuntos com apenas um valor na coluna **tamanho** sofreram permutações aleatórias das observações e foram utilizados 2/3 das mesmas para treinamento e 1/3 para teste.

o tempo de teste - em segundos - e o desvio padrão. Nesta comparação, o modelo ELM obeteve melhor desempenho em grande parte dos casos.

A principal desvantagem sobre a WiSARD, nesse aspecto, se dá na necessidade deste de apresentar o padrão a todos os discriminadores, o que torna o tempo de classificação sensível ao número de classes e ao número de observações de teste, como pode ser observado nos conjuntos de dados *Letter* e *Adult*, respectivamente.

		ELM		WiSARD	
Conjunto	**Tamanho**	**Teste**	**DesvPad**	**Teste**	**DesvPad**
Adult	48842/32561	**1.542**	0.106	170.911	24.843
Australian	690	**0.079**	0.007	1.259	0.230
Banana	400/4900	1.743	0.151	**1.070**	0.213
Diabetes	768	**0.088**	0.006	0.574	0.154
Liver	45	**0.042**	0.005	0.354	0.059
Mushroom	8124	**2.105**	0.141	10.013	0.502

Tabela 4.7: Tempos de teste dos conjuntos de dados binários (em segundos). Resultados obtidos a partir da média de 20 execuções independentes. Conjuntos com apenas um valor na coluna **tamanho** sofreram permutações aleatórias das observações e foram utilizados 2/3 das mesmas para treinamento e 1/3 para teste.

4.4 Uso de Memória

O estudo da utilização de memória RAM foi motivado pela observação feita pelos autores em [28], onde os conjuntos com maior volume de dados tiveram seus experimentos executados em um ambiente computacional diferente, com maior quantidade de memória RAM. O consumo de memória dos modelos podem ser verificados na Tabela 4.9 para os conjuntos de dados binários e na Tabela 4.10. As tabelas apre-

		ELM		WiSARD	
Conjunto	**Tamanho**	**Teste**	**DesvPad**	**Teste**	**DesvPad**
Ecoli	336	**0.040**	0.001	0.755	0.196
Glass	124	**0.027**	0.007	0.590	0.167
Iris	150	**0.018**	0.000	0.070	0.017
Letter	20000	**1.009**	0.051	18.959	0.315
Satimage	4435/2000	**0.660**	0.167	2.640	0.033
Segment	2310	**0.275**	0.018	0.651	0.053
Vehicle	846	**0.102**	0.018	0.480	0.199
Wine	178	**0.021**	0.004	0.210	0.047

Tabela 4.8: Tempos de teste dos conjuntos de dados multiclasse (em segundos). Resultados obtidos a partir da média de 20 execuções independentes. Conjuntos com apenas um valor na coluna **tamanho** sofreram permutações aleatórias das observações e foram utilizados 2/3 das mesmas para treinamento e 1/3 para teste.

sentam a quantidade de memória utilizada pelos principais elementos de cada um dos modelos.

No caso da ELM, foi analisado o custo de se armazenar as matrizes de treino e de teste, a matriz calculada pela pseudoinversa e a matriz de pesos β, enquanto a WiSARD teve medido o tamanho de sua estrutura. De fato, observa-se que o cálculo das matrizes intermediárias de mapeamento, que armazenam todo o conjunto de entrada em formato de ponto flutuante, poderiam gerar problemas de alocação em *hardware* limitado, embora o modelo final da ELM não exija tanta memória.

Como é possível observar, o gasto de memória da WiSARD é consideravelmente menor. É interessante notar que o funcionamento do modelo faz com que o conjunto de entrada binário seja decomposto em fragmentos e fique armazenado em estruturas de dicionário. De certa forma, é como se o gasto de memória fosse composto do tamanho do arquivo de texto binarizado em adição aos *bytes* necessários da estrutura.

		ELM					WiSARD
Conjunto	**Tamanho**	H_{treino}	**pinv**	β	H_{teste}	**Total**	**Total**
Adult	48842/32561	241.296	249.296	0.016	120.480	611.089	**0.066**
Australian	690	3.680	11.680	0.016	1.840	17.217	**0.077**
Banana	400/4900	3.200	11.200	0.016	39.200	53.617	**0.029**
Diabetes	768	4.096	12.096	0.016	2.048	18.257	**0.208**
Liver	45	1.840	9.840	0.016	0.920	12.617	**0.079**
Mushroom	8124	12.000	20.000	0.016	52.992	85.009	**0.419**

Tabela 4.9: Consumo de memória dos conjuntos de dados binários (em mega *bytes*)

		ELM					WiSARD
Conjunto	**Tamanho**	H_{treino}	**pinv**	β	H_{teste}	**Total**	**Total**
Ecoli	336	1.792	9.792	0.064	0.896	12.545	**0.125**
Glass	124	1.136	9.136	0.048	0.576	10.897	**0.113**
Iris	150	0.800	8.800	0.024	0.400	10.025	**0.037**
Letter	20000	106.664	114.664	0.208	53.336	274.873	**3.265**
Satimage	4435/2000	35.480	43.480	0.048	16.000	95.009	**2.146**
Segment	2310	12.320	20.320	0.056	6.160	38.857	**0.464**
Vehicle	846	4.512	12.512	0.032	2.256	19.313	**0.261**
Wine	178	0.944	8.944	0.024	0.480	10.393	**0.187**

Tabela 4.10: Consumo de memória dos conjuntos de dados multiclasse (em mega *bytes*)

Capítulo 5

Conclusões e Trabalhos Futuros

Neste trabalho, foi realizado um estudo comparativo entre dois modelos neurais caracterizados por um baixo tempo de treinamento: a *extreme learning machine* e a WiSARD. Os dois modelos foram implementados em uma mesma linguagem de programação, sendo aplicados a 14 conjuntos de dados públicos com características variadas quanto ao tamanho dos conjuntos, número de atributos e de classes. Além disso, os experimentos foram realizados em um mesmo ambiente computacional. Foram comparadas a acurácia dos modelos, os tempos de treinamento e teste e o uso de memória RAM.

Com os resultados obtidos, pode-se concluir que o modelo WiSARD possui tempo de treinamento inferior ao do modelo ELM, além de exigir menos recursos computacionais. No entanto, o modelo WiSARD sofre perdas de desempenho no tempo de classificação em problemas com número elevado de classes, situação que não apresentou impacto na ELM. Quanto à acurácia dos modelos, embora a ELM apresente resultados nominalmente melhores, a análise do desvio padrão dos experimentos não permite apontar um modelo ganhador visto que apresentaram resultados estatisticamente equivalentes.

No entanto, o trabalho deixa algumas questões ainda não exploradas, sendo oportunidades de futuros trabalhos de pesquisa. Inicialmente, apesar da versatilidade e de ser possível aplicar um processo de desenvolvimento ágil, a linguagem Python ainda possui questões de performance em função de ser uma linguagem interpretada. Desta forma, pretende-se alterar a linguagem de programação utilizada nos experimentos para C++ ou Julia.

Ainda sobre questões de implementação, é válido observar que o modelo WiSARD foi implementado baseando-se no paradigma orientado a objetos, enquanto o modelo ELM foi implementado de forma estruturada. Pretende-se implementar uma nova versão da WiSARD utilizando o paradigma estruturado, sendo possível, assim, realizar uma nova comparação de desempenho entre os modelos, uma vez que o novo paradigma pode levar à redução dos tempos de treinamento e teste.

Um terceiro trabalho envolve a exploração do espaço de configurações. Como é possível observar, a diferença entre a utilização ou não de uma boa configuração mostrou forte impacto nos resultados. Visto que a configuração ideal do modelo ELM acabou por favorecer o mesmo em determinados aspectos, a exploração das configurações ideais do modelo WiSARD para cada um dos conjuntos de dados se mostra necessária para tornar a comparação mais justa. Para conclusão da análise comparativa em um único trabalho, pretende-se também analisar outras funções de ativação para o modelo ELM. Além disso, um estudo aplicando ambos os modelos a outros conjuntos de dados utilizados em [28], além de problemas de aprendizado *online* também pode ser realizado.

Referências Bibliográficas

[1] ABU-MOSTAFA, Y. S., MAGDON-ISMAIL, M., LIN, H.-T., 2012, *Learning From Data*. AMLBook.

[2] ABU-MOSTAFA, Y. S., MAGDON-ISMAIL, M., LIN, H.-T., 2012, *Learning From Data, e-Chapter 7: "Neural networks"*. AMLBook. ISBN: 1600490069, 9781600490064.

[3] ALEKSANDER, I., THOMAS, W., BOWDEN, P., 1984, "WISARD - a radical step forward in image recognition", *Sensor Review*, v. 4, n. 3 (03), pp. 120–124.

[4] ALEKSANDER, I., GREGORIO, M. D., FRANÇA, F. M. G., et al., 2009. "A brief introduction to Weightless Neural Systems". .

[5] BEN-ISRAEL, A., 1980, "Generalized Inverses of Matrices and Their Applications". In: Fiacco, A. V., Kortanek, K. O. (Eds.), *Extremal Methods and Systems Analysis: An International Symposium on the Occasion of Professor Abraham Charnes'*, pp. 154–186, Springer Berlin Heidelberg.

[6] BLEDSOE, W. W., BROWNING, I., 1959, "Pattern Recognition and Reading by Machine". In: *Papers Presented at the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '59 (Eastern), pp. 225–232, New York, NY, USA. ACM.

[7] CARDOSO, D. O., CARVALHO, D. S., ALVES, D. S., et al., 2016, "Financial credit analysis via a clustering weightless neural classifier", *Neurocomputing*, v. 183, pp. 70 – 78. Weightless Neural Systems.

[8] CARDOSO, D. O., FRANÇA, F., GAMA, J. A., 2016, "Clustering Data Streams Using a Forgetful Neural Model". In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, pp. 949–951, New York, NY, USA, . ACM.

[9] CARNEIRO, H. C., FRANÇA, F. M., LIMA, P. M., 2015, "Multilingual Part-of-speech Tagging with Weightless Neural Networks", *Neural Netw.*, v. 66, n. C (jun.), pp. 11–21.

[10] CARVALHO, D. S., CARNEIRO, H. C. C., FRANÇA, F. M. G., et al., 2013, "B-bleaching: Agile Overtraining Avoidance in the WiSARD Weightless Neural Classifier". In: *ESANN*.

[11] CHEN, Y., YAO, E., BASU, A., 2016, "A 128-Channel Extreme Learning Machine-Based Neural Decoder for Brain Machine Interfaces", *IEEE Transactions on Biomedical Circuits and Systems*, v. 10, n. 3 (June), pp. 679–692.

[12] CHOI, K., TOH, K.-A., BYUN, H., 2012, "Incremental face recognition for large-scale social network services", *Pattern Recognition*, v. 45, n. 8, pp. 2868 – 2883.

[13] COUTINHO, P. V. S., CARNEIRO, H. C. C., CARVALHO, D. S., et al., 2014. "Extracting rules from DRASiW's "mental images"". .

[14] DE AGUIAR, K., FRANÇA, F. M. G., BARBOSA, V. C., et al., 2015, "Early detection of epilepsy seizures based on a weightless neural network". pp. 4470–4474.

[15] DE SOUZA, D. F. P., FRANÇA, F. M. G., LIMA, P. M. V., 2015, "Real-Time Music Tracking Based on a Weightless Neural Network". pp. 64–69.

[16] DO NASCIMIENTO, D. N., LIMA DE CARVALHO, R., MORA-CAMINO, F. A. C., et al., 2015, "A WiSARD-based multi-term memory framework for online tracking of objects". pp. 978–287587014–8, Bruges, Belgium, abr.

[17] FENG, C., SUTHERLAND, A., KING, R., et al., 1993, "Comparison of machine learning classifiers to statistics and neural networks". In: *Proceedings of the Third International Workshop in Artificial Intelligence and Statistics*, pp. 41–52.

[18] G-B HUANG, H. WANG, Y. L., 2011, "Extreme learning machines: a survey", *Int J Mach Learn Cybern*, p. 107–122.

[19] GOLDSCHMIDT, R., 2010, *Introdução à Inteligência Computacional - Fundamentos, ferramentas e aplicações*. Disponível em: <`www.boente.eti.br/boente2012/fuzzy/ebook/ebook-fuzzy-goldschmidt.pdf`>.

[20] GREGORIO, M. D., GIORDANO, M., 2016, "Cloning DRASiW systems via memory transfer", *Neurocomputing*, v. 192, pp. 115 – 127.

[21] GRIECO, B. P., LIMA, P. M., GREGORIO, M. D., et al., 2010, "Producing pattern examples from "mental" images", *Neurocomputing*, v. 73, n. 7–9, pp. 1057 – 1064.

[22] GUANG-BIN HUANG, L. C., 2007. "Convex incremental extreme learning machine". .

[23] GUANG-BIN HUANG, L. C., SIEW, C.-K., 2006. "Universal Approximation Using Incremental Constructive Feedforward Networks With Random Hidden Neurons". .

[24] HUANG, G.-B., 2013. "Extreme Learning Machines". Disponível em: <`http://www.ntu.edu.sg/home/egbhuang/index.html`>.

[25] HUANG, G.-B., 2014. "An insight into Extreme Learning Machines: random neurons, random features and kernels". .

[26] HUANG, G.-B., 2015. "What are Extreme Learning Machines? Filling the gap between Frank Rosenblatt's dream and John von Neumann's puzzle". .

[27] HUANG, G.-B., CHEN, L., 2007, "Convex incremental extreme learning machine", *Neurocomputing*, v. 70, n. 16–18, pp. 3056 – 3062.

[28] HUANG, G.-B., ZHOU, H., DING, X., et al., 2012, "Extreme Learning Machine for Regression and Multiclass Classification", *Trans. Sys. Man Cyber. Part B*, v. 42, n. 2 (abr.), pp. 513–529. ISSN: 1083-4419.

[29] IGELNIK, B., IGELNIK, B., ZURADA, J. M., 2013, *Efficiency and Scalability Methods for Computational Intellect*. 1st ed. Hershey, PA, USA, IGI Global.

[30] JAEGER, H., 2001, "The ''echo state'' approach to analysing and training recurrent neural networks - with an Erratum note", .

[31] KASUN, L. L. C., YANG, Y., HUANG, G. B., et al., 2016, "Dimension Reduction With Extreme Learning Machine", *IEEE Transactions on Image Processing*, v. 25, n. 8 (Aug), pp. 3906–3918.

[32] KHAKI, K., STONHAM, T. J., 2012, "Face Recognition with Weightless Neural Networks Using the MIT Database". pp. 228–233, Berlin, Heidelberg, Springer Berlin Heidelberg.

[33] LI, M.-B., HUANG, G.-B., SARATCHANDRAN, P., et al., 2005, "Fully complex extreme learning machine", *Neurocomputing*, v. 68, pp. 306 – 314.

[34] LICHMAN, M., 2013. "UCI Machine Learning Repository". Disponível em: <http://archive.ics.uci.edu/ml>.

[35] MAASS, W., NATSCHLÄGER, T., MARKRAM, H., 2002, "Real-time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations", *Neural Comput.*, v. 14, n. 11 (nov.), pp. 2531–2560.

[36] MARTÍNEZ-MARTÍNEZ, J. M., ESCANDELL-MONTERO, P., SORIA-OLIVAS, E., et al., 2011, "Regularized extreme learning machine for regression problems", *Neurocomputing*, v. 74, n. 17, pp. 3716 – 3721.

[37] OLIVEIRA, L. F. R., FRANÇA, F. M. G., 2017, "ELM vs. WiSARD: a performance comparison", Submitted.

[38] SCHRAUWEN, B., VERSTRAETEN, D., CAMPENHOUT, J. V., 2007, "An overview of reservoir computing: theory, applications and implementations". In: *ESANN*, pp. 471–482.

[39] SURI, M., PARMAR, V., 2015, "Exploiting Intrinsic Variability of Filamentary Resistive Memory for Extreme Learning Machine Architectures", *IEEE Transactions on Nanotechnology*, v. 14, n. 6 (Nov), pp. 963–968.

[40] SUYKENS, J., VANDEWALLE, J., 1999, "Least Squares Support Vector Machine Classifiers", *Neural Processing Letters*, v. 9, n. 3, pp. 293–300.

[41] THEANO DEVELOPMENT TEAM, 2016, "Theano: A Python framework for fast computation of mathematical expressions", *arXiv e-prints*, v. abs/1605.02688 (maio). Disponível em: <http://arxiv.org/abs/1605.02688>.

[42] VAPNIK, V., 1998, "The Support Vector Method of Function Estimation". In: Suykens, J. A. K., Vandewalle, J. (Eds.), *Nonlinear Modeling: Advanced Black-Box Techniques*, pp. 55–85, Boston, MA, Springer US.

[43] WANG, Y., YU, H., NI, L., et al., 2015, "An Energy-Efficient Nonvolatile In-Memory Computing Architecture for Extreme Learning Machine by Domain-Wall Nanowire Devices", *IEEE Transactions on Nanotechnology*, v. 14, n. 6 (Nov), pp. 998–1012.

[44] ZENG, Y., XU, X., FANG, Y., et al., 2015, "Traffic Sign Recognition Using Deep Convolutional Networks and Extreme Learning Machine". pp. 272–280, Cham, Springer International Publishing.

Apêndice A

Exploração do Espaço de Configurações da WiSARD

Este apêndice trata da exploração do espaço de configurações do modelo WiSARD. Embora uma exploração mais geral se faça necessária, julga-se apropriado iniciar seu estudo. Desta forma, apresenta-se um método geral para exploração do modelo aplicado a um conjunto de dados qualquer.

A.1 Metodologia

Como apontado anteriormente, tanto a ELM quanto a WiSARD possuem alguns parâmetros para serem configurados: para a primeira, o número de neurônios e o fator de regularização C a ser adicionado no cálculo da pseudo-inversa, enquanto a segunda necessita do tamanho da tupla e o número de *bits* atribuído a cada atributo do conjunto de dados.

Os experimentos com a ELM utilizaram as configurações informadas em [28], enquanto os experimentos com a WiSARD foram feitos utilizando uma mesma configuração para todos os conjuntos. Foram realizados testes utilizando os mesmos valores para o tamanho das tuplas e para o número de bits de cada atributo. Os valores foram {10, 20, 30, 40, 50}, como mostrados na Tabela A.1 e na Tabela A.2.

Como é possível observar, a configuração afeta cada conjunto de dados de maneira diferente. No entanto, continua sendo necessário estudar a sensibilidade de cada atributo de maneira mais específica. Desta forma, nesta seção são apresentados os procedimentos a serem utilizados para a descoberta das configurações ideais de um modelo WiSARD quando utilizado em um conjunto de dados qualquer. Para fins ilustrativos, será utilizado o conjunto *Banana*, uma vez que o mesmo possui apenas dois atributos.

Visto que o valor inicial 20 obteve um melhor desempenho geral para o conjunto

# *bits*	Estatística	Adult	Australian	Banana	Diabetes	Liver	Mushroom
10	Acurácia	0.77	0.65	0.80	0.70	0.57	**0.86**
	Treino	0.95	0.02	0.59	0.35	0.01	5.30
	Teste	475.66	0.07	10.19	0.32	0.16	12.19
20	Acc	**0.79**	**0.71**	**0.89**	0.67	0.55	**0.86**
	Treino	1.87	0.03	0.94	0.46	0.02	8.74
	Teste	179.58	0.04	2.78	0.17	0.10	15.93
30	Acc	0.77	0.65	0.88	0.71	0.53	**0.86**
	Treino	2.22	0.04	1.16	0.63	0.02	11.96
	Teste	123.26	0.03	1.42	0.42	0.16	21.98
40	Acc	0.78	0.66	0.88	**0.72**	0.53	**0.86**
	Treino	3.10	0.06	1.53	0.81	0.03	14.87
	Teste	114.24	0.04	1.35	0.52	0.21	27.45
50	Acc	0.78	0.67	0.86	0.69	0.51	**0.86**
	Treino	3.27	0.07	1.85	0.96	0.03	18.03
	Teste	77.45	0.08	1.21	1.07	0.27	32.35

Tabela A.1: Exploração inicial do espaço nos conjuntos de dados binários. A primeira coluna indica o número de *bits* atribuído a cada um dos atributos do conjunto de dados, sendo seguido das estatísticas de acurácia média, tempo de treinamento e tempo de teste do conjunto utilizando a respectiva configuração Resultados obtidos através da média de 20 execuções independentes utilizando validação cruzada com 10 subconjuntos.

# bits	Estatística	Ecoli	Glass	Iris	Letter	Satimage	Segment	Vehicle	Wine
10	Acurácia	0.79	**0.87**	0.93	0.60	0.71	0.87	0.59	0.95
	Treino	0.01	0.01	0.00	1.54	0.69	0.19	0.06	0.01
	Teste	0.01	0.01	0.00	24.92	4.51	0.20	0.04	0.00
20	Acc	0.81	0.85	**0.94**	0.82	0.86	0.93	**0.70**	**0.98**
	Treino	0.02	0.01	0.01	2.70	1.14	0.29	0.10	0.02
	Teste	0.02	0.01	0.00	8.70	3.74	0.22	0.05	0.01
30	Acc	**0.84**	0.86	0.93	0.90	**0.89**	0.95	**0.70**	0.91
	Treino	0.03	0.01	0.01	3.46	1.57	0.41	0.14	0.02
	Teste	0.04	0.02	0.00	9.57	4.39	0.31	0.06	0.01
40	Acc	0.82	0.82	0.88	**0.93**	**0.89**	**0.96**	**0.70**	0.82
	Treino	0.03	0.02	0.01	4.02	2.10	0.54	0.19	0.03
	Teste	0.06	0.05	0.01	11.17	5.62	0.40	0.10	0.02
50	Acc	0.80	0.82	0.82	**0.93**	**0.89**	**0.96**	0.67	0.69
	Treino	0.03	0.03	0.01	5.21	2.44	0.64	0.21	0.03
	Teste	0.07	0.06	0.01	14.34	9.07	0.62	0.12	0.03

Tabela A.2: Exploração inicial do espaço nos conjuntos de dados multiclasse. A primeira coluna indica o número de *bits* atribuído a cada um dos atributos do conjunto de dados, sendo seguido das estatísticas de acurácia média, tempo de treinamento e tempo de teste do conjunto utilizando a respectiva configuração. Resultados obtidos através da média de 20 execuções independentes utilizando validação cruzada com 10 subconjuntos.

analisado, este foi o valor utilizado como base de exploração. Foi escolhido o intervalo (15,25) para o número de bits de cada atributo, o que totaliza 121 combinações. Foram, então, realizados experimentos utilizando todas as combinações dos valores nesse intervalo para os dois atributos, sendo medidas a acurácia e os tempos de treino e teste de cada uma utilizando validação cruzada de 10 subconjuntos.

Também é necessário observar que para cada combinação do número de bits dos atributos existe um conjunto de valores referentes aos possíveis tamanhos de tupla que podem ser escolhidos. Esse conjunto é dado pelos divisores do valor resultante da soma do número de *bits* de cada atributo. Por exemplo, se o experimento em questão utilizar 15 *bits* para cada atributo - isto é, o experimento (15,15) -, será criada uma palavra de 30 *bits* e o conjunto de possíveis tuplas é dado por {1, 2, 3, 5, 6, 10, 15, 30}, de forma que cada tamanho de tupla afeta as três estatísticas analisadas. Desta forma, cada um dos 121 experimentos precisou ser executado mais de uma vez para que todos os tamanhos de tupla pudessem ser analisados.

A.2 Algoritmo

O processo descrito na seção anterior é representado pelo Algoritmo 2. O algoritmo recebe como entrada os parâmetros **limiteInferior** e **limiteSuperior**, que serão utilizados para definir os valores mínimos e máximos que cada atributo terá durante a exploração. O algoritmo inicia definindo a estrutura **melhorResultado**, que armazena três valores de interesse na análise: a acurácia do experimento, o tempo de treinamento e o tempo de teste. Esta estrutura armazenará o melhor dentre todos os resultados do experimento. Em seguida, é gerada uma lista com todas as combinações possíveis de valores entre o limite inferior e superior para todos os atributos, iniciando-se então um *loop* onde cada configuração é testada.

Dentro do *loop*, declara-se a estrutura **melhorLocarl** que, semelhante à estrutura anterior, armazenará o melhor dentre os resultados dos experimentos seguindo a configuração vigente. O algoritmo segue calculando o valor total da soma dos valores de *bits* da configuração, gerando então a lista com todos os divisores possíveis. Inicia-se então um segundo *loop* que percorre todos os possíveis tamanhos de tuplas.

O resultado do experimento é armazenado na estrutura **resultadoLocal**, sendo comparada em seguida com o melhor resultado. Se a acurácia - armazenada no primeiro índice da estrutura - for melhor do que a melhor local, substitui-se o valor de **melhorLocal** pelo encontrado. Saindo do *loop*, verifica-se se o melhor resultado local foi melhor do que o global. Caso positivo, substitui-se o valor.

```
Dados: limiteInferior, limiteSuperior
Resultado: Melhor configuração encontrada
início
    melhorResultado = [0,0,0]
    listaConfiguracoes = gerarCombinacoes(numeroParametros,
     limiteInferior, limiteSuperior)
    para cada i em listaConfiguracoes faça
        melhorLocal = [0,0,0]
        total = soma(listaConfiguracoes[i])
        listaDivisores = divisores(total)
        para k = 1, ..., tamanho(listaDivisores) faça
            resultadoLocal = WiSARD(listaConfiguracoes[i], k)
            se resultadoLocal[0] > melhorLocal[0] então
                melhorLocal = resultadoLocal
            fim
        fim
        se melhorLocal[0] > melhorResultado[0] então
            melhorResultado = melhorLocal
        fim
    fim
fim
```

Algoritmo 2: Algoritmo para exploração do espaço de configurações

A.3 Análise

A Figura A.1 apresenta a superfície formada pela variação da acurácia final do modelo frente às possíveis combinações de tamanhos de termômetro em cada atributo. É possível observar que não há uma variação significativa para o conjunto de dados em questão, fato já indicado pela tabela de resultados iniciais. A Figura A.2 apresenta uma versão ampliada dessa superfície.

A Figura A.3 apresenta a superfície formada pela variação nos tempos de treinamento. É possível notar um crescimento muito próximo ao linear conforme os dois parâmetros aumentam, o que é plausível, dado que o tamanho da palavra de entrada aumenta gradativamente, o que faz com que o tempo de processamento das subpalavras também aumente.

Por fim, a Figura A.4 apresenta a superfície formada pela variação dos tempos de teste. Pode-se observar a redução do tempo conforme os dois parâmetros aumentam, embora não de forma tão suave como o tempo de treinamento. Considerando que o valor apresentado no gráfico é o relacionado à melhor configuração de tamanho de tupla e que tuplas de tamanho maior são processadas mais rapidamente, é possível concluir que, conforme os atributos aumentam de tamanho, divisões de tuplas maiores tendem a obter melhores resultados de acurácia, resultando também na redução do tempo de teste, embora gere aumento no tempo de treinamento.

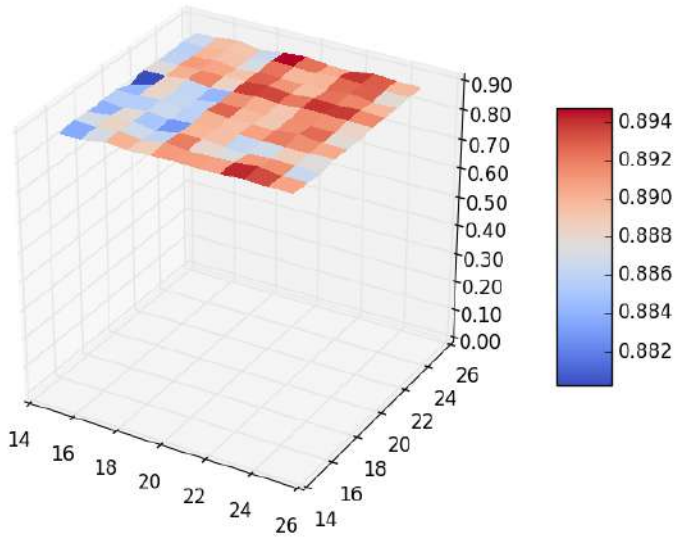

Figura A.1: Acurácia durante a exploração do espaço de configurações

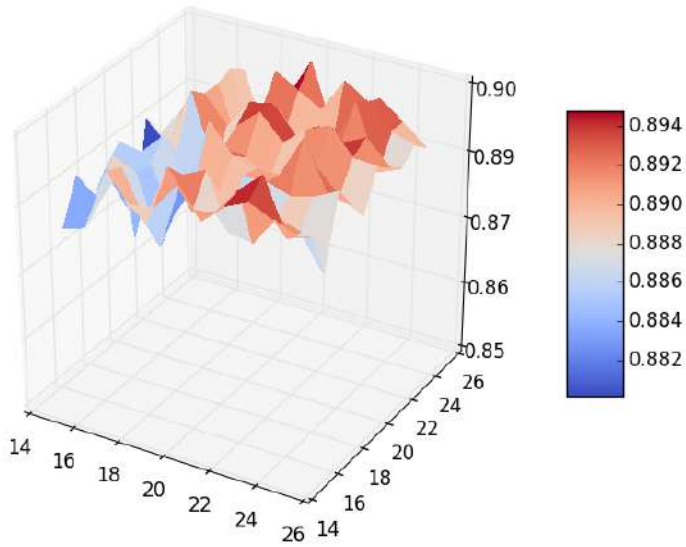

Figura A.2: Acurácia durante a exploração do espaço de configurações

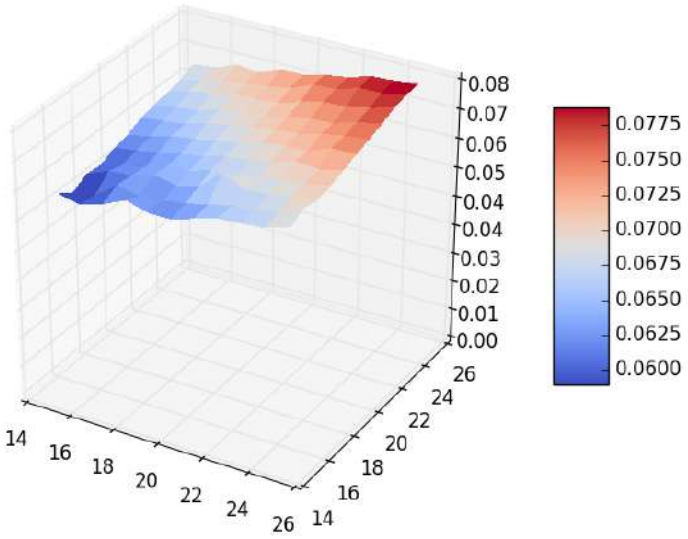

Figura A.3: Tempo de treino durante a exploração do espaço de configurações

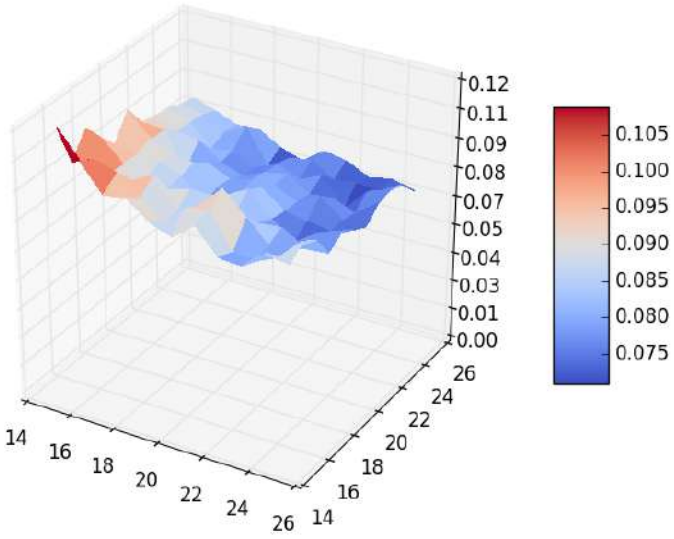

Figura A.4: Tempo de teste durante a exploração do espaço de configurações

Apêndice B

Artigo aceito para publicação

Neste apêndice está incluso o artigo submetido e aceito para apresentação no 25º Simpósio Europeu de Redes Neurais Artificiais, Inteligência Computacional e Aprendizado de Máquina (ESANN 2017). O trabalho apresenta resultados iniciais que levaram ao desenvolvimendo desta dissertação.

ELM vs. WiSARD: a performance comparison

Luiz Fernando R. Oliveira and Felipe M. G. França *

Systems Engineering and Computer Science Program, COPPE
Universidade Federal do Rio de Janeiro, RJ, Brazil

Abstract. The extreme learning machine (ELM) is known for being a fast learning neural model. This work presents a performance comparison between ELM and the WiSARD weightless neural network model, regarding training and testing times, and classification accuracy as well. The two models were implemented in the same programming language and experiments were carried out on the same hardware environment. By using a group of datasets from the public repositories UCI and Statlog, experimental results shows that the WiSARD presented training times approximately one order of magnitude smaller than ELM, while classification accuracy varied according the number of classes involved. However, while WiSARD's architecture setups were not exhaustively searched, architecture setups for ELM were kept the same as the ones found in the literature as the best for each given dataset.

1 Introduction

There has been great interest on the improvement of more recent and sophisticated techniques, regarding both learning theory and implementation aspects. A typical example is deep learning, which has gathered large attention due to GPU computing advances, in order to deal with the reduction of it's huge training time. However, training time is still an issue when trying to apply machine learning into online situations. Two neural models oriented to high performance training are **Wi**lkes, **S**tonhan and **A**leksander **R**ecognition **D**evice (WiSARD) and the extreme learning machine (ELM). The first is a Weightless neural network model that uses RAM-based neurons and a set of discriminators with pseudo-random input mapping, while the second is a single layer feedfoward network in which the hidden layer do not need to be iteractively adjusted.

These two models share interesting properties like low training time and ease of implementation, while having some particularities, such as WiSARD being able to be implemented directly in Boolean logic hardware and ELM in a single-step iteration. Motivated by these properties, this work proposes a comparative study of these two techniques regarding training time, testing time and average classification accuracy when applied to several datasets having different characteristics.

*The authors would like to thank CAPES - Ministry of Education, Brazil, CNPq - Ministry of Science and Technology, Brazil - grant number 307437/2013-2, FINEP - Ministry of Science and Technology, Brazil.

2 Extreme Learning Machines

Classical neural network models, such as multi-layer perceptrons, are usually formed by an input layer, a set of hidden layers and an output layer. The hidden layers contains weights that need to be optimized by an algorithm like backpropagation. ELMs intend to be fast trainable by using just a single hidden layer with random nodes that do not need to be tuned. This is done by generating a random feature mapping matrix $H_{N\times L}$, where N is the number of data samples and L is the number of neurons. H is defined as:

$$H = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_L, b_L, x_1) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_N) & \cdots & G(a_L, b_L, x_N) \end{bmatrix},$$

where $a_i \in R^d$ and $b_i \in R^+$ are random parameters, x_i is a sample of dataset $X = (x_i, t_i)|x_i \in R^d, t_i \in R^m, i = 1, ..., N$, where m is the number of attributes involved, and $G(a_i, b_i, x)$ is the output function of the ith hidden node.

As presented in [4], the basic ELM algorithm consists in randomly generating the hidden nodes parameters (a_i, b_i), create the hidden output matrix H and calculate the output weight vector β by solving $\beta = H^\dagger T$, where $H^\dagger$ is the *Moore-Penrose generalized inverse*. Usually, the orthogonal projection method is preferable to generate this inverse, and is defined as $H^\dagger = (H^T H)^{-1} H^T$ if $H^T H$ is nonsingular or $H^\dagger = H^T (HH^T)^{-1}$ if HH^T is nonsingular.

It is also suggest to add a positive value $1/C$ to the diagonal of the matrix that is being inverted in order to improve the generalization performance, which results in the modification of the algorithm so that $\beta = (\frac{I}{C} H^T H)^{-1} H^T T$ or $\beta = H^T (\frac{I}{C} HH^T)^{-1} T$

3 The WiSARD weightless neural network

Proposed by Wilkes, Stonhan and Aleksander in 1984 [8], WiSARD is a weightless neural network model which aims at recognizing patterns represented as binary data. It's a multidiscriminator model where each discriminator is in charge of recognizing a specific class of patterns. A discriminator is formed by a set of m RAM-like structures based on a pseudo-random mapping of the input data field [3] [7].

Figure 1 shows how a discriminator is trained. Initially, all RAM structures stores 0 in all of its it's addressable contents. Given a binary input, m groups of n bits are randomly defined. The combination of the values in each element creates a memory address, and the value 1 is then stored at the address of the corresponding RAM. [6]

Figure 2 shows how a pattern is classified. A pattern is presented to all discriminators and, for each one, the addresses bits are selected the same way as the training by using the discriminator pseudo-random mapping. Each RAM

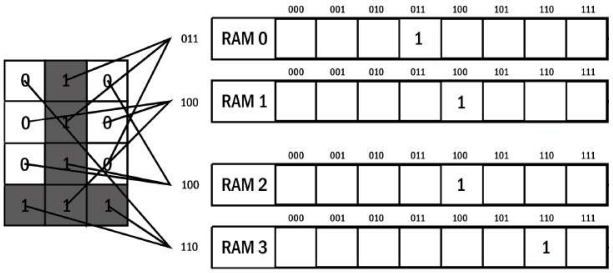

Fig. 1: Training a WiSARD discriminator

is verified using the created addresses and, if the value stored if different from 0, the RAM is said to be "activated".

At the end of the verification, each discriminator returns a value r, related to it's activation response. The discriminator with highest response implies in the classification value of the presented pattern. Figure 2 also show that the discriminator related to class **1** presents the highest response, meaning that the input pattern was classified as being a **1**.

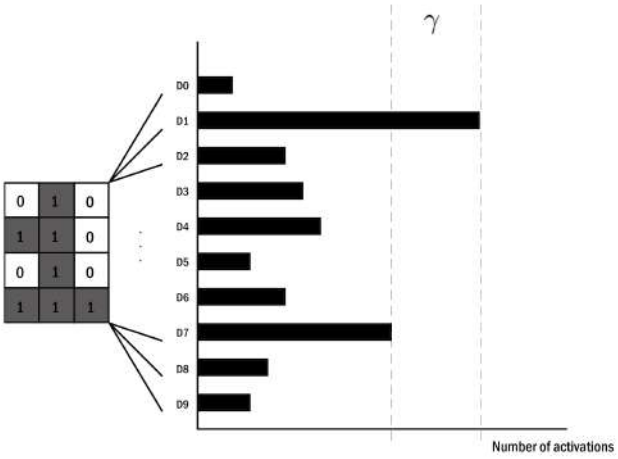

Fig. 2: Classifying a pattern using WiSARD.

4 Experimental Framework

4.1 Datasets

To compare both models, a subset of the datasets used in [5] was selected. Most of these datasets were obtained from the UCI public repository [1] and present very different characteristics, regarding number of samples, number of attributes, as well as type of attributes. A total of 14 datasets were used, six of them being binary classification datasets, while eight other are multiclass problems. Some of the datasets were already divided into training and testing sets, while others consist on a single data file. For the latest, two different methodologies were used. The first is a 10-fold cross validation and the second is the same used in [5]: 2/3 of data for training and 1/3 for testing in combination with a random permutation of data. Following the setup defined in [5], data was normalized between [-1, 1] when exercising the ELM model.

4.2 Architectural parameters

For the ELM model, it was used the *sigmoid additive node*, while the number of neurons and the regularization factor C were the ones specified in [5] that optimizes the model stabilization. These parameters were obtained by the author by exploring the combination space. As such exploration is yet to be done in WiSARD, it was decided that the binarization of the data values would be done by using a 20 bit thermometer representation for each feature of the sample. Thus, the configuration would result in a $m \times 20$ retina, and a tuple of size $n = 20$ was used in all of the experiments.

4.3 Environment

Both models were implemented using the Python 2.7 language on a Intel Core i3 2.7GHz CPU with 16GB of RAM memory and Ubuntu Linux 14.04 operating system.

5 Results and Discussion

Table 1 shows the experimental results for both binary and multiclass datasets. The table shows the test accuracy using both methodologies, as well as training and testing time performance of both models. All the results were obtained through the mean value after 20 independent runs. The standard deviation of both accuracies were significantly small, varying within the range (0.009, 0.054).

By observing the testing time of WiSARD on some of the datasets, it is possible to observe that some of them are far slower than the one of ELM. Two factors can be pointed out: (i) the first is related to the number of discriminators; the greater the number of classes a dataset has, greater number of comparisons the WiSARD needs to perform; (ii) the second, which also relates to the first, is related to the network's architecture; If the input field is big and tuples are

Table 1: Binary (1-6) and multiclass (7-14) datasets; Accuracy 1 (10-fold cross validation); Accuracy 2 ([5]); best values in boldface.

Dataset	Model	Accuracy 1	Accuracy 2	Training	Testing
1-Adult	ELM	**0.811**	**0,802**	8,160 s	**1,542** s
	WiSARD	0.802	0,773	**0,329** s	170,911 s
2-Australian	ELM	**0.778**	**0,739**	0,817 s	**0,079** s
	WiSARD	0.715	0,687	**0,246** s	1,259 s
3-Banana	ELM	**0.876**	**0,877**	0,682 s	1,743 s
	WiSARD	0.856	0,848	**0,029** s	**1,070** s
4-Diabetes	ELM	**0.778**	**0,765**	0,888 s	**0,088** s
	WiSARD	0.677	0,706	**0,366** s	0,574 s
5-Liver	ELM	**0.687**	**0,722**	0,849 s	**0,042** s
	WiSARD	0.554	0,570	**0,125** s	0,354 s
6-Mushroom	ELM	0.463	**0,859**	1,379 s	**2,105** s
	WiSARD	**0.855**	0,850	**0,298** s	10,013 s
7-Ecoli	ELM	0.766	**0,876**	0,781 s	**0,040** s
	WiSARD	**0.805**	0,811	**0,143** s	0,755 s
8-Glass	ELM	0.805	**0,899**	0,717 s	**0,027** s
	WiSARD	**0.854**	0,875	**0,116** s	0,590 s
9-Iris	ELM	**0.966**	**0,974**	0,618 s	**0,018** s
	WiSARD	0.943	0,935	**0,038** s	0,070 s
10-Letter	ELM	0.574	**0,932**	6,205 s	**1,009** s
	WiSARD	**0.819**	0,808	**1,339** s	18,959 s
11-Satimage	ELM	0.573	**0,882**	2,567 s	**0,660** s
	WiSARD	**0.854**	0,848	**0,950** s	2,640 s
12-Segment	ELM	**0.938**	**0,962**	1,346 s	**0,275** s
	WiSARD	0.923	0,933	**0,284** s	0,651 s
13-Vehicle	ELM	0.683	**0,828**	0,792 s	**0,102** s
	WiSARD	**0.696**	0,674	**0,634** s	0,480 s
14-Wine	ELM	0.901	**0,980**	0,791 s	**0,021** s
	WiSARD	**0.972**	0,952	**0,128** s	0,210 s

small, more RAM structures will be created, raising even more the number of comparisons.

6 Conclusion

This work presented a comparison between two classification models that are characterized by fast training. Interestingly, ELM turned out to provide a higher accuracy on all of the datasets using the second methodology, which used the parameters presented in [5], although WiSARD could perform in a similar way on the majority of them. Meanwhile, the 10-fold crossvalidation presented a variation on accuracies, which reinforces the argument that the best architec-

tural parameters of the models can drastically affect the accuracy performance. Nevertheless, as stated before, WiSARD's best configurations are yet to be explored.

Regarding training time, the WiSARD presented better training performance in all datasets. As for testing time, it should be stated that the WiSARD configuration implies directly on this component of the model. Although WiSARD presented a similar classification time in the majority of the datasets, there were a small group it was far slower than ELM. It is important to notice that, for bigger datasets, ELM can present a memory usage issue. It is stated in [5] that for the four largest datasets, a different computer with higher amount of RAM memory was used. But WiSARD has presented no issues regarding memory usage. Thus, it turns to be necessary to add this comparison in a future work. Finally, although ELM can also be applied to regression problems, it's yet to be discussed the applications of weightless neural network models in this kind of domain, which is another motivation for future studies.

References

[1] C. L. Blake and C. J. Merz, "UCI Repository of Machine Learning Databases," Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, 1998.

[2] D. S. Carvalho, H. C. C. Carneiro, F. M. G. França and P. M. V. Lima, "B-bleaching: Agile Overtraining Avoidance in the WiSARD Weightless Neural Classifier." ESANN (2013).

[3] F. M. G. França, M. De Gregorio, P. M. V. Lima, W. R. Oliveira Jr, "Advances in Weightless Neural Systems". Proc. of ESANN 2014. Brussels: i6doc.com, 2014. p. 497-504.

[4] G-B Huang, H. Wang, Y. Lan, Extreme learning machines: a survey. Int J Mach Learn Cybern 2(2):107–122, 2011

[5] G-B Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme Learning Machine for Regression and Multiclass Classification". Trans. Sys. Man Cyber. Part B 42, 2 (April 2012), 513-529.

[6] H. L. França, J. C. P. Silva, M. De Gregorio, O. Lengerke, M. S. Dutra, F. M. G. França, "Movement Persuit Control of an Offshore Automated Platform via a RAM-based Neural Network". In: 11th. Int. Conf. Control, Automation, Robotics and Vision, 2010, Singapore.

[7] I. Aleksander, M. De Gregorio, F. M. G. França, P. M. V. Lima, H. Morton, "A Brief introduction to Weightless Neural Systems". Proc. of ESANN 2009. Evere, Belgium: d-side, 2009. p. 299-305.

[8] I. Aleksander, W. Thomas, and P. Bowden, "WiSARD: a radical step forward in image recognition," Sensor review, vol. 4, no. 3, pp. 120– 124, 1984.

[9] K. M. Khaki, "Weigthless Neural Networls for Face and Pattern Recognition: an Evaluation using open-source databases". PhD thesis, Brunel University, 2013.