UMA ARQUITETURA PARA AGRUPAMENTO DE DADOS EM FLUXO CONTÍNUO BASEADA EM REDES NEURAIS SEM PESOS

Douglas de Oliveira Cardoso

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

Fevereiro de 2012

UMA ARQUITETURA PARA AGRUPAMENTO DE DADOS EM FLUXO CONTÍNUO BASEADA EM REDES NEURAIS SEM PESOS

Douglas de Oliveira Cardoso

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. Adilson Elias Xavier, D.Sc.

Prof. Antonio de Pádua Braga, Ph.D.

Prof. Josefino Cabral Melo Lima, Docteur

RIO DE JANEIRO, RJ – BRASIL

FEVEREIRO DE 2012

Cardoso, Douglas de Oliveira

Uma Arquitetura para Agrupamento de Dados em Fluxo Contínuo Baseada em Redes Neurais Sem Pesos/Douglas de Oliveira Cardoso. – Rio de Janeiro: UFRJ/COPPE, 2012.

XII, 47 p.: il.; 29, 7cm.

Orientador: Felipe Maia Galvão França

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2012.

Referências Bibliográficas: p. 45 – 47.

1. Redes neurais sem pesos. 2. Agrupamento de dados. 3. Mineração de dados em fluxo contínuo. I. França, Felipe Maia Galvão. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Filho, aprenda o que eu lhe ensino e nunca esqueça o que mando você fazer. Escute os sábios e procure entender o que eles ensinam. Sim, peça sabedoria e grite pedindo entendimento. Procure essas coisas, como se procurasse prata ou um tesouro escondido. Se você fizer isso, saberá o que quer dizer temer o Senhor, e aprenderá a conhecê-lo.
(Provérbios 2:1 - 5)

Ao meu avô Ney.

Agradecimentos

Agradeço a Deus, Senhor de tudo. Se cheguei onde estou é porque Ele quis. Eu louvo ao Senhor porque Ele é bom.

Agradeço á minha esposa, Marceley, por toda a paciência que teve comigo. Nos momentos em que a paciência acabou, demos um jeito, certo?

Aos meus pais, Rogerio e Rosana, pelo investimento em mim. Quanto mais velhor fico mais vejo o amor de vocês principalmente nas mais duras broncas.

A minha irmã Kenia, avós Ney, Ilda, Nonô e Belinha, tios, primos e tantos outros familiares. Somos inseparáveis, e assim, insuperáveis.

A tantos que me fortaleceram nos Projetos Vida Nova de Jardim América e Vigário Geral. Trabalhamos muito, e vamos comemorar muito também.

Aos companheiros de tantas batalhas acadêmicas, desde todos da saudosa Info 042 áqueles com quem desenvolvi parcerias durante o meu mestrado. Aprendi com vocês muito mais do que com livros.

Àqueles com quem trabalhei no CECAFA e DAbM, militares e civis.

Ao professor Felipe, meu orientador, cujo investimento para o desenvolvimento desse trabalho foi muito além de conhecimentos de computação. Também cito aqui os professores João Gama e Priscila Lima, com quem espero repetir as parcerias de sucesso muito em breve.

A todos que fazem do PESC o melhor programa de pós-graduação em computação do país. Sou grato a todos os professores e demais funcionários desta unidade pela contribuição na minha formação acadêmica.

Agradeço ao CNPq, AROSA e COPPETEC pelo suporte financeiro.

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

UMA ARQUITETURA PARA AGRUPAMENTO DE DADOS EM FLUXO CONTÍNUO BASEADA EM REDES NEURAIS SEM PESOS

Douglas de Oliveira Cardoso

Fevereiro/2012

Orientador: Felipe Maia Galvão França

Programa: Engenharia de Sistemas e Computação

O agrupamento de dados em fluxo é um tema de pesquisa relativamente novo, o qual adiciona mais restrições ao problema de agrupamento clássico: os grupos são definidos num instante no tempo, considerando dados mais recentes uma maior influência na formação dos grupos que dados mais antigos. Existem outros desafios como processar os dados num ritmo mais veloz que aquele no qual eles chegam, e capturar pequenas mudanças durante a análise do fluxo de dados. StreamWiSARD, uma arquitetura para agrupamento de dados em fluxo contínuo apresentada nessa dissertação buscar atender a todos os requerimentos do problema ainda tendo um baixo consumo de memória devido ao pequeno número de microgrupos mantido durante seu funcionamento, enquanto a maioria das alternativas tenta gerenciar o maior número possível de microgrupos. Além disso, é mostrado um novo mapeamento útil para adaptar dados expressos em valores reais para serem introduzidos a classificadores baseados na similaridade binária das suas entradas. Os experimentos conduzidos mostram algumas propriedades interessantes consequentes do uso de discriminadores WiSARD como unidades básicas de sintetização de dados.

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

A WEIGHTLESS NEURAL NETWORK-BASED ARCHITETURE FOR STREAM DATA CLUSTERING

Douglas de Oliveira Cardoso

February/2012

Advisor: Felipe Maia Galvão França

Department: Systems Engineering and Computer Science

Stream data clustering is relatively new research theme, which adds more constraints to the classical clustering problem: the clusters are defined for an instant in time, considering more recent data a stronger influence in the clusters definition than older data. There are other challenges as to process data in a faster pace than this of its arrival, and to capture even small change events during the stream data analysis. StreamWiSARD, a stream data clustering system presented in this dissertation looks to fulfil all problem requirements while having a low memory footprint due to the small number of microclusters maintained during its execution, while the majority of other options try to always manage as much microclusters as possible. Besides this, it is shown a new mapping useful to adapt real-valued data to be input to classifiers which work based on binary similarity. The experiments conducted show some interisting properties consequent to the use of WiSARD discriminators as the basic data synthesizing unit.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A cada dia a quantidade de dados existentes no mundo cresce, e muito. Diferente de outras épocas nas quais uma minoria destacada fornecia informação a uma maioria, qualquer um utilizando um simples telefone celular é capaz de se tornar uma fonte de informação. Dados obtidos de um único indivíduo talvez tenham pouco valor quando considerados isoladamente, mas a combinação de vários pequenos pedaços pode levar a fatos até então não notados, como se o todo formado fosse maior que a soma das partes que o compõe. A mineração de dados ocupa um papel-chave nesse contexto, sendo uma área de pesquisa para o desenvolvimento de métodos para extração de informação relevante, de qualidade, de quantidades em geral enormes de dados.

A capacidade de identificar grupos de elementos semelhantes é uma habilidade cognitiva básica, usada pelo ser humano em diversas situações do seu dia-a-dia. Desenvolver sistemas computacionais que hajam como se tivessem tal habilidade é um dos desafios da área de mineração de dados, principalmente considerando que um dos objetivos é extrapolar a habilidade humana em certos pontos, processando enormes quantidades de informação e identificando relações que muitas vezes não perceberíamos "manualmente" inspecionando dados.

O agrupamento de dados em fluxo contínuo é um tema de pesquisa relativamente recente, e que traz novas restrições ao problema clássico de agrupamento de dados. É requerido de uma abordagem para essa nova versão do problema grande rapidez

no processamento dos dados assim como a capacidade de responder rapidamente a mudanças nos dados como o surgimento e o desaparecimento de classes de instâncias. Ainda é preciso que esta seja capaz de lidar com dados descritos em um grande número de dimensões ou contendo ruído. São diversas dificuldades, que reforçam a consideração de alto grau de complexidade do problema. Será apresentada nessa dissertação a nossa contribuição para a pesquisa nesse tema.

1.1 Objetivos e contribuições

O principal objetivo dessa dissertação é apresentar o StreamWiSARD, uma arquitetura para agrupamento de dados em fluxo contínuo que tem como elementos básicos discriminadores WiSARD. Outra novidade a ser apresentada aqui é o mapeamento desenvolvido para que dados inicialmente representados como vetores reais n-dimensionais pudessem ser apresentados aos discriminadores. Ainda falaremos da realização de *bleaching* beaseado em um critério temporal, sendo a forma escolhida para implementar o comportamento de esquecimento de dados considerados antigos, um dos requerimentos para o problema aqui abordado.

A primeira contribuição deste trabalho está no próprio desenvolvimento do StreamWiSARD. É um modelo simples, e com baixíssimos requerimentos computacionais para seu funcionamento. Indo na direção contrária do desenvolvimento de outros sistemas de igual finalidade, o StreamWiSARD busca produzir bons resultados mantendo o menor número possível de microgrupos sendo gerenciado durante seu funcionamento, enquanto outros objetivam gerenciar uma quantidade imensa destes. Será mostrado que é possível conseguir ótimos resultados ainda utilizando poucos recursos.

O mapeamento desenvolvido aqui poderia ser facilmente replicado em outros sistemas onde fosse desejado utilizar discriminadores WiSARD ou qualquer outro dispositivo para classificação baseado em entradas binárias e sua similaridade. Além disso, contribuímos para a discussão sobre a questão da representação dos dados para seu processamento, o que é um tema muitas vezes subvalorizado.

Por fim, o uso do *bleaching* segundo critérios temporais aumenta a já vasta lista de critérios para a realização dessa operação em discriminadores WiSARD. Além disso, esta parte do trabalho está diretamente ligada à realização de agrupamento de dados em fluxo baseado no paradigma de janela deslizante, quando o paradigma geralmente usado é o de janela amortizada. Isso é interessante visto que não há superioridadeentre ambos, mas apenas diferenças que tornam em uma certa situação o uso de um sistema para agrupamento de dados em fluxo contínuo mais ou menos conveniente de acordo com o paradigma sobre o qual o mesmo está fundamentado.

1.2 Estrutura da dissertação

No Capítulo 2 é apresentado o tema 'agrupamento em fluxos contínuos', sendo apresentados alguns sistemas e paradigmas importantes dentro deste. No Capítulo 3 é feita uma introdução às redes neurais sem pesos, falando da sua inspiração biológica e dos modelos WiSARD e AutoWiSARD. Esses são os temas que integrados formam o cenário no qual este trabalho está inserido. No Capítulo 4 nosso trabalho é apresentado, sendo devidamente relacionado com os trabalhos que serviram de inspiração para este e apresentando as novas ideias introduzidas para que chegássemos ao produto final obtido. No Capítulo 5 colocamos à prova a arquitetura desenvolvida, comparando-a com outros sistemas com o mesmo propósito, explicitando diferenças entre eles. No Capítulo 6 são apresentadas as conclusões às quais chegamos até esse momento na pesquisa e indicamos algumas possíveis direções de continuidade da mesma.

Capítulo 2

Agrupamento em fluxos de dados

A mineração de dados é um tema de pesquisa em computação para o desenvolvimento de técnicas, algoritmos e métricas para a obtenção de informação relevante e o reconhecimento de padrões em massas de dados. A mineração de fluxos de dados é um subtema da mesma. Uma aplicação enquadrada no primeiro tipo é também do segundo caso possua a seguinte característica: a massa de dados sendo processada não está completamente disponível *a priori*, mas é gradativamente fornecida, com o decorrer do tempo. Exemplos desse tipo de conjunto de dados são os pacotes que trafegam numa rede de computadores num período ou uma sequência de leituras de uma rede de sensores.

Enfim, o agrupamento em fluxos de dados é uma das tarefas mais comuns dentro do subtema supracitado, que consiste na definição e manutenção de grupos (*clusters*) de dados conforme critérios de similaridade entre elementos da entrada. Numa tarefa de agrupamento "clássico" todo o conjunto de dados é analisado e são definidos grupos considerando igualmente todos os pontos do conjunto. Porém, numa tarefa envolvendo fluxos de dados o conjunto de grupos deve ser atualizado iterativamente, conforme a entrada é processada. Geralmente os dados mais recentes são considerados mais relevantes para a definição dos grupos, segundo parâmetros da tarefa. Ou seja, os dados mais antigos devem ser "esquecidos" com o tempo. A evolução de grupos de dados é ilustrada na Figura 2.1.

Segundo a descrição do conjunto de dados fornecida é possível chegar a certas

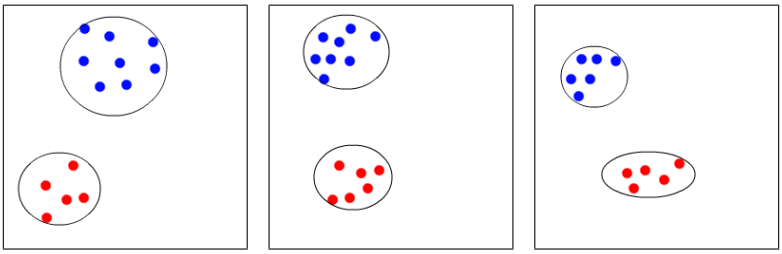


Figura 2.1: Um exemplo da evolução dos grupos de um conjunto de dados, numa sequência de momentos

conclusões. Eis algumas:

- O conjunto pode ser arbitrariamente grande, o que impede que este seja completamente armazenado para ser processado. Por isso é considerado ideal que cada instância apresentada seja processada uma única vez e em seguida descartada, ou então devem ser definidos critérios para o armazenamento que garantam que este não será feito irrestritamente;

- A velocidade com que os dados são disponibilizados pode ser altíssima, o que força o rápido processamento destes e, consequentemente, serve como agravante para o armazenamento de grandes porções da entrada para posterior uso no tratamento das instâncias. Caso haja lentidão, devido ao acúmulo de instâncias a serem processadas, partes da entrada podem ser descartadas indesejadamente.

Apresentamos a seguir alguns trabahos de pesquisa sobre o tema os quais serviram como referência para o desenvolvimento deste. Posteriormente tratamos especificamente de modelos existentes para lidar com a temporalidade dos dados.

2.1 CluStream

O CluStream [3] é uma arquitetura desenvolvida para a tarefa de agrupamento em fluxos de dados que ocupa um lugar de destaque na história da pesquisa no assunto. Isso se deve ao fato deste introduzir alguns conceitos e elementos que foram posteriormente utilizados em diversos outros métodos.

Um desses conceitos é a definição de duas fases de funcionamento da arquitetura:

- A fase *online*, que é executada continuamente, processando a entrada conforme o surgimento de novas de instâncias. Ela é responsável por manter uma estrutura que reflita aquilo que foi processado até um certo momento, atendendo a critérios como esquecimento gradual e rapidez de funcionamento, conforme já comentado;

- A fase *offline*, que utiliza a estrutura mantida pela fase *online* para definir os grupos que a arquitetura apresenta como saída. Esta pode ser executada em qualquer momento durante o processamento do fluxo de dados. Ela não possui nenhum requerimento quanto à rapidez, visto que não lida diretamente com a entrada.

Outro conceito, uma contribuição-chave deste trabalho, é o microgrupo (*microcluster*), que é uma estrutura que sintetiza estatisticamente informações de um conjunto de instâncias, servindo como representante do conjunto durante o funcionamento da arquitetura. Os microgrupos servem como uma camada intermediária de sintetização dos dados, sendo a camada final os próprios grupos definidos como saída da arquitetura. Resumidamente, a componente *online* da arquitetura funciona determinando para cada instância recebida o microgrupo mais próximo a ela, segundo a distância euclidiana até o centro do microgrupo, para atualizá-lo conforme as características da mesma. Logo, o uso dos microgrupos permite condensar as instâncias em conjuntos segundo a similaridade destas, como que diminuindo a resolução da entrada. O microgrupo é o menor grão da arquitetura. A Figura 2.2 apresenta graficamente o conceito do microgrupo.

O uso do CluStream pode ser problemático em situações onde os grupos não são esféricos, pois este utiliza o K-means [4] na fase *offline* para agrupar os microgrupos, formando os grupos. Isso impede o uso da arquitetura em diversos problemas reais. Além disso, para o uso do K-means é necessário informar o número de grupos a serem definidos, sendo que esse número pode variar consideravelmente durante o

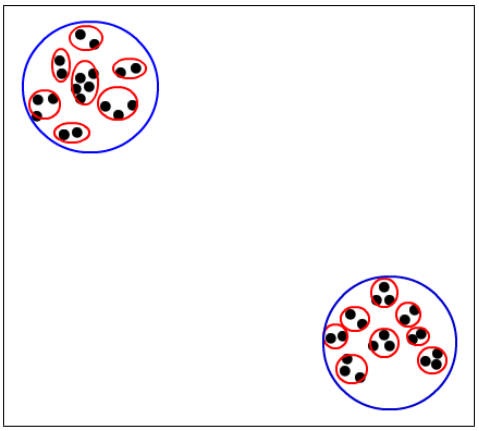


Figura 2.2: Uma ilustração da relação entre dados (preto), microgrupos (vermelho) e grupos (azul)

processamento do fluxo de dados. Ou seja, seria interessante que esse número fosse automaticamente ajustado.

2.2 ClusTree

Outro importante método para agrupamento em fluxos de dados é o ClusTree [5]. Como seu nome sugere, o algoritmo trabalha baseado em uma árvore, que é balanceada e na qual cada nó é um microgrupo: os nós internos sintetizam as características de seus filhos, enquanto as folhas, das instâncias a elas relacionadas. Assim, é possível encontrar o microgrupo mais próximo de uma instância sem que todos os existentes sejam considerados como acontece no CluStream: basta seguir um caminho da raiz até uma das folhas guiando-se nos nós internos da árvore pelas características neles armazenadas. Fica evidente então a primeira qualidade marcante desse método: a rapidez com a qual pode processar cada instância, pela complexidade logarítmica da operação de busca pelo microgrupo mais próximo.

Também interessante sobre o algoritmo é o fato dele ser um *anytime algorithm* [6]. Ou seja, ele é capaz de auto-ajustar o seu funcionamento de acordo com o tempo disponível para cada operação. Por exemplo, digamos que no tratamento de uma instância, durante o percurso da raiz até uma das folhas outra instância é recebida, sendo que esta será descartada caso uma terceira surja. Então o algoritmo possui mecanismos para encerrar o processamento em andamento, aproveitando a parte do

caminho já percorrida, para que possa então atender a instância recém-chegada.

Outra peculiaridade do ClusTree é a falta de definição de um procedimento para a fase *offline* do algoritmo. Isto foi feito assim pois segundo os seus desenvolvedores qualquer método de agrupamento "clássico" poderia ser utilizado: este receberia como entrada os microgrupos armazenados nas folhas da árvore, definindo assim os grupos, a saída do algoritmo.

2.3 DenStream, D-Stream e MR-Stream

Assim como o CluStream [3] tem características que remetem ao K-Means [4], inclusive fazendo uso do próprio na sua fase *offline*, o DenStream [7] inspira-se no DBSCAN [8]: na sua fase *online* faz a manutenção de microgrupos que servem de entrada para uma versão modificada do DBSCAN que é a componente *offline* do algoritmo. O conceito de conectividade-por-densidade do DBSCAN ganhou novas características aqui devido a característica temporal dos dados: dois microgrupos considerados conectados-por-densidade (*density-connected*) num dado instante de tempo podem não estar mais nesse estado quando avaliados num instante de tempo posterior, devido a possibilidade do desaparecimento gradual de microgrupos envolvidos nessa ligação.

O D-Stream [9] utiliza uma estrutura de dados chamada grade de densidade (*density grid*) para aglomerar os dados em primeira instância: todo o espaço é dividido em células hipercúbicas de mesma medida, sendo cada instância recebida mapeada para uma dessas células. Conforme as células são alimentadas, sua densidade é atualizada para que na fase *offline* do algoritmo os grupos sejam definidos conforme a conectividade-por-densidade destas. Um dos pontos mais fortes deste algoritmo é a mínima complexidade no processamento das instâncias: a célula apropriada para uma instância é identificada numa complexidade constante, graças a divisão regular do espaço. Mas essa divisão também tem uma desvantagem: o algoritmo mantém o mesmo nível de resolução em todo o espaço, ainda que apenas determinadas áreas deste apresentem exemplos num dado intervalo de tempo.

Por fim, o MR-Stream [10] apresenta-se como uma evolução do D-Stream, também dividindo o espaço em células, mas de maneira recursiva como numa Quad-tree [11]. O uso dessa estrutura de dados no formato de uma árvore torna o MR-Stream mais próximo do próprio ClusTree: o processamento de uma instância de ambos tem, no pior caso, uma complexidade logarítmica, mas o MR-Stream utiliza uma tabela *hash* para agilizar a identificação de uma célula mais adequada para uma dada instância caso essa célula já esteja em uso. Além disso, é capaz de ajustar a resolução usada em cada região do espaço, o que o D-Stream era incapaz de fazer.

2.4 Mecanismos de esquecimento de dados

Existem diversas formas de implementação do comportamento de esquecimento gradativo de dados antigos de acordo com o recebimento de novos. Abordamos aqui 2 padrões que se destacam tanto pela popularidade de uso no desenvolvimento de trabalhos no tema aqui abordado quanto pela abrangência de situações reais capazes de serem neles modeladas. A apresentação de cada um deles e uma comparação de ambos são feitas a seguir.

- **Janela amortizada** [12] (*damped window*). Neste, cada instância possui um peso, uma indicação de sua importância temporal. Este peso decresce exponencialmente com o passar do tempo. A função mais usada para indicar o peso de objetos em aplicações baseadas neste modelo é:

$$\omega(\Delta t) = \beta^{-\lambda \Delta t}$$

  Segundo esta função, num dado instante de tempo t um objeto recebido no instante t' tem peso $\omega(t - t')$. O fator de decaimento λ serve para controlar a rapidez de esquecimento dos dados. O valor mais comum de β é 2, o que resulta numa meia-vida dos objetos de $1/\lambda$;

- **Janela deslizante** [12]. Aqui, dado um horizonte, só são considerados para a

saída do algoritmo objetos recebidos num instante posterior ao horizonte indicado. Qualquer objeto recebido num instante anterior ao horizonte indicado, *i.e.* fora da janela temporal, deve ser totalmente desconsiderado. Em outras palavras, o peso dos objetos é binário: 1, para todos dentro da janela, e 0, para aqueles fora dela.

O modelo de janela amortizada é utilizado pela maioria dos algoritmos para agrupamento de dados em fluxo contínuo citados nesse capítulo. Isto se deve a maior facilidade de implementação desse, sendo apresentado em trabalhos como Kranen *et al.* [5] mecanismos simples através dos quais é possível facilmente atualizar o peso de microgrupos conforme esses recebem novas instâncias.

É interessante notar que nesse modelo, as instâncias perdem importância com o tempo mas, teoricamente, nunca são descartadas. Por outro lado, no modelo de janela deslizante o limite é bem definido. Não há superioridade em um dos 2 tipos, mas uma melhor aplicabilidade de um ou de outro dependendo da situação.

Capítulo 3

Redes neurais sem pesos

Uma rede neural artificial (RNA) é um modelo computacional originalmente inspirado em redes neurais biológicas (RNB). Seu objetivo básico é reproduzir características das RNB como aprendizado, capacidade de inferência e reconhecimento de padrões. Por isso, de maneira simplificada, as RNA imitam a estrutura das RNB, sendo compostas por neurônios (ou nós) interligados de alguma forma. Ainda que certos tipos de nós sejam realmente simples, a combinação de vários elementos simples pode levar a formação de estruturas capazes de realizar tarefas complexas. A Figura 3.1 apresenta um possível esquema de ligações entre neurônios formando uma estrutura com duas entradas e uma saída.

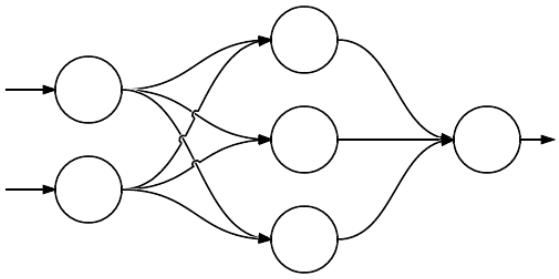


Figura 3.1: Um possível esquema de conexões entre neurônios

3.1 Redes neurais e os pesos

Uma estrutura das mais comuns de RNA é formada por diversos nós com apenas uma saída, calculada como um somatório dos valores fornecidos nas suas entradas, interligados por sinapses. Cada uma dessas sinapses possui um peso, que modifica o valor da saída de um neurônio que será utilizado na entrada de outro ao qual ele está conectado. Por exemplo, a saída de um neurônio pode ser multiplicada pelo peso da sinapse, sendo este resultado usado como uma das entrada de outro neurônio. Para esse tipo de RNA a adequação da rede para a tarefa que esta deve realizar é feita através do ajuste dos pesos das sinapses. Ou seja, o conhecimento é armazenado não nos nós, nos elementos básicos, mas nas conexões entre eles, as sinapses. Esse tipo de modelagem de um neurônio biológico foi desenvolvida por Warren McCulloch e Walter Pitts.

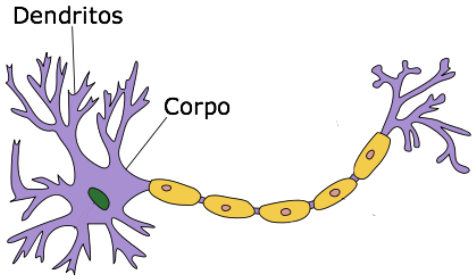


Figura 3.2: Um neurônio biológico [1]

Agora observe as Figuras 3.2 e 3.3. Os estímulos, a entrada, de um neurônio biológico são recebidos através dos dendritos, que formam um estrutura no formato de árvore. Essa estrutura determina a existência de dendritos mais influentes que outros, de acordo com a proximidade deste para o corpo do neurônio: caso um dendrito do tipo inibitório receba um estímulo, todos os estímulos recebidos por outros dendritos abaixo dele na árvore serão inibidos, ignorados. No caso de um estímulo excitatório, ele então reafirma o estímulo de sua sub-árvore. Em suma, das várias árvores dendríticas chegam sinais ao corpo do neurônio para que seja então gerado um sinal em resposta a estes. Do ponto de vista computacional, isso é um processo de decodificação de uma entrada binária numa saída pré-definida.

Neurônios artificiais baseados nesse princípio de funcionamento, claramente diferente do neurônio de McCulloch-Pitts, são utilizados nas redes neurais sem pesos (RNSP). Nestas, o conhecimento é armazenado nos nós, que geram saídas definidas de acordo com estímulos, entradas, previamente recebidos. Não há nenhum tipo de pesos nas sinapses, que apenas repassam os estímulos produzidos por neurônios.

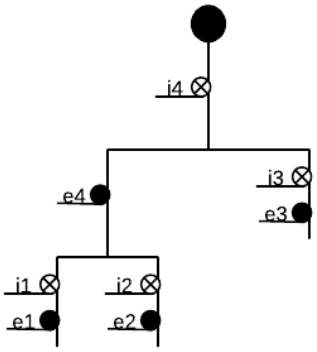


Figura 3.3: Uma representação de uma árvore dendrítica, com extremidade excitatórias (e) e inibitórias (i).

3.2 WiSARD

O WiSARD (Wilkie, Stonham & Aleksander's Recognition Device) [13] é um tipo de RNSP. Originalmente foi usada no reconhecimento de imagens em preto-e-branco similares. Por isto, o formato básico da entrada deste é uma cadeia de bits, o que pode forçar a definição de uma representação binária de objetos que não estejam originalmente nesse padrão. Nas subseções adiante gradualmente seu funcionamento será explicado.

3.2.1 De imagens para endereços de memória

Buscando uma forma simples de entender o funcionamento do WiSARD, consideremos a comparação de figuras binárias, *i.e.* em preto-e-branco, descritas numa grade $n \times m$: dado conjunto de figuras, é possível intuitivamente afirmar que as duas

com o maior número de bits em comum formam o par mais similar do conjunto. Considere agora, que para evitar fazer nm comparações entre bits para determinar o quão parecidas duas figuras são, vamos compara-las linha-por-linha. Observe que este critério é mais rigoroso: no primeiro, bit-a-bit, caso duas figuras tivessem n bits diferentes, elas teriam uma semelhança relativa de $\frac{nm-n}{nm}$ %; na comparação linha-a-linha, caso cada um dos n bits diferentes que elas possuem estejam distribuídos nas n linhas, seria correto afirmar que a semelhança relativa entre elas é de 0%, já que nenhuma das suas linhas seria igual. Dessa maneira é possível ajustar o nível de rigor do critério de similaridade usado no WiSARD, através da definição do tamanho dos grupos de bits nos quais as entradas são divididas.

Depois de ser estabelecido o tamanho dos grupos de bits nos quais as entradas serão quebradas, ainda está em aberto por quais bits cada grupo será formado além da ordem dos mesmos no grupo. Chamaremos de mapeamento a entidade que indica com que bits cada grupo é composto. Estabeleceremos três fatos importantes quanto a esse mapeamento:

- Ele define grupos de tamanhos iguais, o que garante que nenhum deles é considerado mais importante que o outro. Numa situação extrema, se os bits de duas figuras $n \times m$ fossem divididos em 2 grupos, um com apenas 1 bit e o outro com o restante, seria possível afirmar que a semelhança relativa entre elas é de 50% se apenas esse bit que forma esse grupo unitário tivesse o mesmo valor em ambas, o que pode ser bem diferente da verdade. Pelo mesmo princípio cada bit faz parte de um grupo apenas;

- Ele é definido aleatoriamente, objetivando assim descorrelacionar a entrada, por exemplo, evitando que bits inicialmente próximos e por isso com maior chance de terem o mesmo valor nas instâncias processadas sejam usados na formação dos endereços para o mesmo nó RAM, o que tornaria parte da informação de tais endereços, e consequentemente do nó, redundante;

- Ele não só define a que grupo cada bit pertencerá mas também define a ordem

dos bits nos grupos. Sendo assim, é possível codificar um grupo de bits nessa certa ordem num número inteiro, conforme já visto.

Tendo sido definido um mapeamento, cada figura binária é introduzida num sistema WiSARD antes passando por este. Os grupos definidos pelo mapeamento serão utilizados no endereçamento dos nós RAM. A figura 3.4 ilustra esse acontecimento.

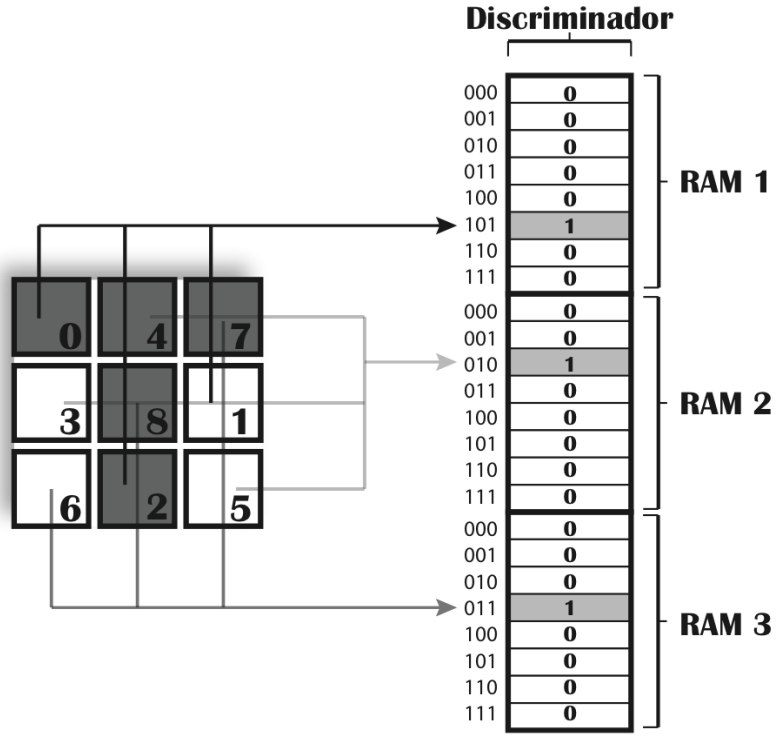


Figura 3.4: O mapeamento de uma figura 3 × 3 em 3 endereços dos nós RAM [2]

3.2.2 Discriminadores

Geralmente um sistema baseado no WiSARD é composto por um grupo de discriminadores. Um discriminador é uma estrutura que no seu uso básico é responsável pelo reconhecimento de uma única classe dentre aquelas a serem diferenciadas. O esquema de um discriminador poder ser visto na Figura 3.5. Conforme pode ser visto, ele é composto por vários nós RAM. Para processar uma instância, na cadeia de bits apresentada na entrada é aplicado o já definido mapeamento, o que leva a formação de um vetor de n números referentes a cada grupo de bits, sendo n o número de nós RAM do discriminador. Cada um desses números servirá para

indicar um endereço em um nó de um discriminador.

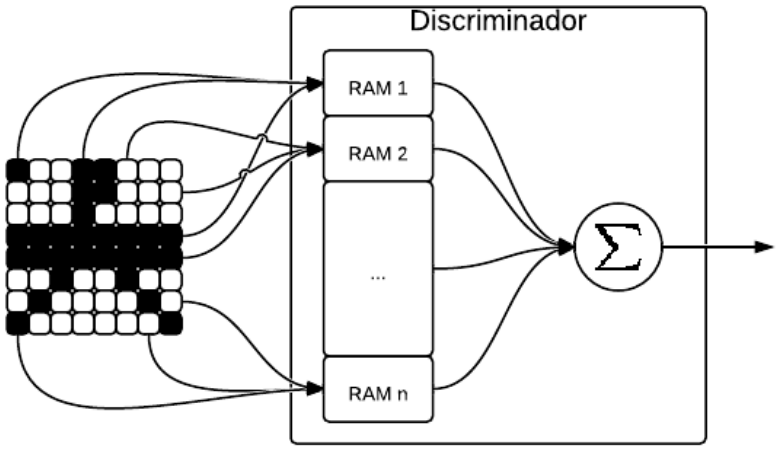


Figura 3.5: O esquema de um discriminador

Caso a instância sendo processada esteja sendo aprendida, cada posição das RAMs indicada pelos endereços determinados é marcada, estando inicialmente todas as posições desmarcadas. Mas, se a situação for a avaliação do reconhecimento da instância pelo discriminador, cada RAM produz uma saída binária: '1', se a posição indicada pelo endereço por ela recebido estava marcada, ou '0', caso contrário. O discriminador dá como resposta a essa avaliação o somatório das saídas dos seus nós RAM.

Num sistema multidiscriminador, ou seja, que serve para o reconhecimento de instâncias de entrada entre várias classes, a instância é considerada da classe representada pelo discriminador cuja resposta for a mais alta entre todos. Isso é ilustrado na Figura 3.6. Na Figura o sistema julga a entrada como sendo da classe '4'. O discriminador referente a classe '9' apresenta a segunda melhor resposta, o que é coerente com o que podemos observar: há uma considerável semelhança entre as classes '4' e '9'.

3.2.3 *Bleaching*

Um dos problemas aos quais o WiSARD está sujeito ocorre quando um discriminador aprende um número muito grande de instâncias da classe a qual ele representa e, devido às pequenas diferenças entre tais instâncias, gradualmente passa a reco-

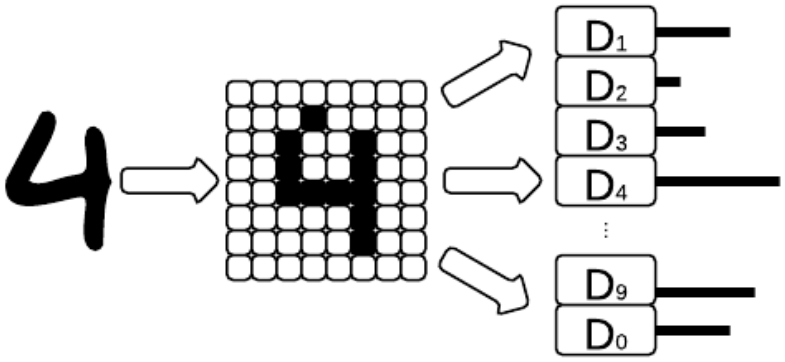


Figura 3.6: Um exemplo de sistema multidiscriminador de números manuscritos.

nhecer instâncias de outra classe. Usando o exemplo daSeção anterior, devido ao aprendizado de uma grande quantidade de exemplos do número '4', o discriminador referente a essa classe poderia cada vez mais e mais ter altas pontuações de reconhecimento de instâncias da classe '9'. Esse problema é chamado saturação, pelo estado em que geralmente se encontram os nós RAM dos discriminadores nessa situação: com a maioria das suas posições marcadas.

Mas num certo nó RAM saturado devido a ter sido muito treinado, é provável que algumas posições tenham sido escritas várias vezes, enquanto outras foram escritas esporadicamente. Novamente voltando ao exemplo numérico, cada pessoa escreve o número '4' de forma única, mas somos capazes de reconhecer um '4' quando o vemos devido a um padrão, a um formato mínimo ao qual esperamos que um caracter numérico qualquer que acreditamos ser um '4' atenda. É coerente acreditar que as posições marcadas mais vezes nos nós RAM de um discriminador sejam referentes a esse formato básico da classe representada por ele, visto que todas as instâncias da classe compartilhariam dessa base.

Se gradualmente apagassemos as posições menos gravadas dos nós de um discriminador, ou seja, aquelas que foram gravadas devido a peculiaridades de instâncias, mas que não representam o cerne da classe, estaríamos combatendo diretamente a saturação. Esse processo é chamado *bleaching*. Para que este aconteça é definido um limiar, uma frequência mínima de escrita de para que uma posição de um nó RAM não seja limpa. Além disso, é necessária uma pequena modificação nos nós RAM dos discriminadores: ao invés de cada posição guardar apenas 1 bit indicando

se já foi marcada ou não, elas passam a guardar um contador, indicando quantas vezes já foram marcadas.

Outro efeito positivo no uso do *bleaching* é que este também minimiza os efeitos do treinamento de um discriminador por um pequeno número de instâncias de uma classe diferente da classe que ele representa. O raciocínio é o mesmo: pela provável baixa frequência de escrita das posições marcadas quando essas instâncias foram absorvidas, elas seriam atingidas pela filtragem feita.

3.3 AutoWiSARD

O AutoWiSARD [14] é um modelo de rede neural inspirado no WiSARD, tendo discriminadores deste em sua estrutura. O propósito desse modelo, porém, é o agrupamento de imagens binárias segundo sua similaridade, e não classificação das mesmas.

Há no funcionamento deste modelo um certo grau de dinamismo na gerência nos discriminadores: eles são criados durante o processamento do conjunto de dados. A criação de discriminadores é determinada por uma janela de aprendizado implementada pelo modelo (Figura 3.7). Para explicar seu funcionamento, consideremos o processamento de uma instância do conjunto de dados num sistema AutoWiSARD. Na primeira parte dessa ação é identificado o discriminador o qual melhor reconhece essa instância. Seja ele d_{max} e sua resposta r_{max}. Sendo r_{best} a melhor resposta possível de um discriminador, ou seja, o número de nós RAM que este possui e w_{min} e w_{max} ($0 \leq w_{min} \leq w_{max} \leq r_{best}$) os parâmetros da janela, há três possiblidades para o que acontece a seguir:

- Se $r_{best} \leq w_{min}$, um novo discriminador é criado o qual aprende essa instância. Aqui, foi considerado que nenhum dos discriminadores existente representava a classe dessa figura;

- Se $w_{min} < r_{best} \leq w_{max}$, é feita uma opção entre duas ações probabilisticamente. Seja $p = \dfrac{r_{best} - w_{min}}{w_{max} - w_{min}}$. A probabilidade de criação de um novo

discriminador é $1 - p$. A probabilidade de treinamento parcial de d_{max} pela instância é p;

- Se $w_{max} < r_{max}$, o processamento da instância era encerrado, sendo considerado que já existe um discriminador o qual representa bem a classe da mesma, não havendo necessidade de treinamento deste.

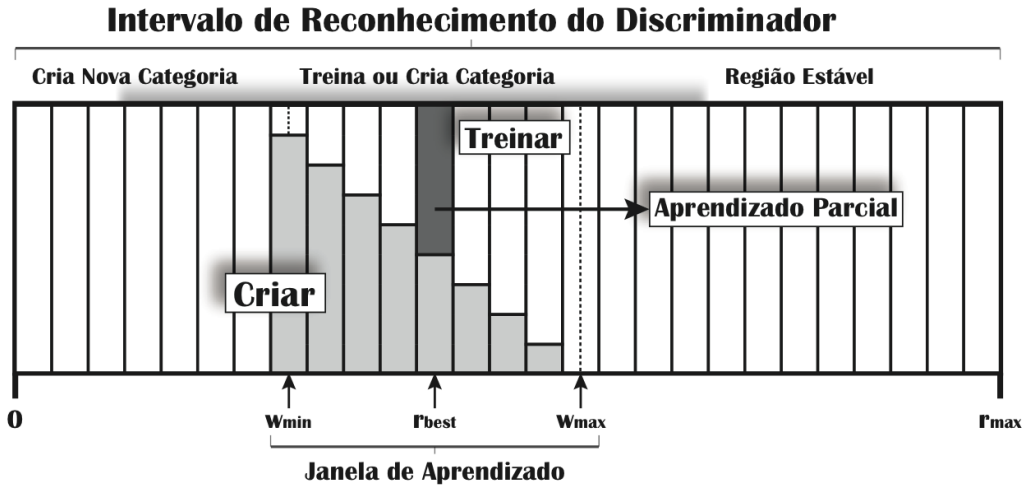


Figura 3.7: Janela de aprendizado do AutoWiSARD [2]

O treinamento parcial é uma interessante característica do AutoWiSARD. No treinamento normal, todos os nós RAM do discriminador tem uma de suas posições marcadas. Aqui são marcados apenas $w_{max} - r_{max}$ nós RAM escolhidos aleatoriamente entre aqueles que não responderam positivamente quando a instância sendo processada foi apresentada ao discriminador. Repare que essa estratégia combate diretamente a saturação do discriminador, treinando o mínimo de nós RAM possível para que este pudesse reconhecer a instância.

Capítulo 4

StreamWiSARD

Neste Capítulo, vamos explicar este método, desenvolvido neste trabalho. Proporcionando um breve resumo, a arquitetura é estruturado em duas componentes, de forma semelhante ao CluStream[3], onde a parte *online* continuamente processa os dados de entrada mantendo aquilo que servirá de entrada para a parte *offline*. As seções 4.2 e 4.3 são dedicadas à explicação de cada uma dessas partes.

De certo ponto de vista, a definição do microgrupo mais adequado para absorver uma certa instância do fluxo de dados é um processo de classificação. Diferente de outros métodos, que baseiam-se em estruturas simples para sintetizar em primeira instância os dados processados, aqui foi considerado interessante averiguar as consequências do uso de uma estrutura com maior poder de aprendizado para essa função: o discriminador WiSARD.

As instâncias de entrada são representadas como vetores multidimensionais reais. Como a parte *online* da arquitetura baseia-se no uso de discriminadores WiSARD, que lidam exclusivamente com entradas binárias, em primeiro lugar é necessário entender como nós representamos um exemplo desta forma. A Seção 4.1 apresenta esta parte do trabalho.

Visando facilitar a compreensão do trabalho, é apresentado na Figura 4.1 um esquema completo das entidades do mesmo, além da indicação do tipo de informação que elas trocam quando se comunicam. Cada elemento presente nesse diagrama será devidamente introduzido no decorrer deste capítulo.

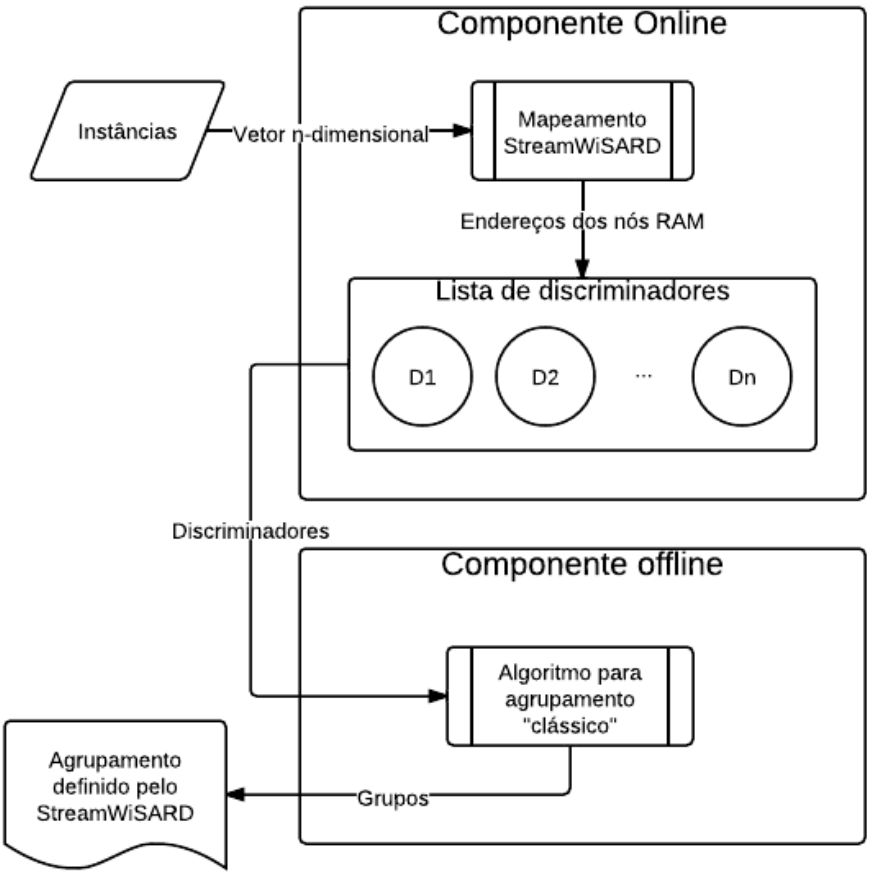


Figura 4.1: Uma representação esquemática do StreamWiSARD

4.1 De vetores reais para endereços de memória

Obviamente, existem diversas maneiras de codificar um vetor n-dimensional real em uma cadeia binária. Uma das estratégias mais simples e diretas de fazer isto é converter cada um dos n valores reais do vetor em uma cadeia binária e concatená-las. Considerando que os valores desse vetor hipotético são normalizados no intervalo $[0, 1]$ e têm uma precisão de a dígitos, é possível representar qualquer valor usando $b = \lceil \log_2 10^a \rceil$ bits. Consequentemente, um exemplo poderia ser representado em uma cadeia de bits de comprimento $l = bn$.

Lembrando como o WiSARD funciona, procuraramos um processo de codificação que converta dois vetores reais n-dimensionais em cadeias binárias tais que o número de bits diferentes entre elas é proporcional à dissemelhança entre os vetores. Apenas a utilização de uma representação apropriada dos valores em cada dimensão poderia deslocar a representação completa para este objetivo, já que a conversão do valor

de um vetor em uma dada dimensão é feito de forma independente dos valores nas outras dimensões. Nesta representação, o número de bits diferentes de duas cadeias de bits que representam 2 escalares reais deverá ser proporcional à própria diferença entre os escalares. O *Binary Reflected Gray Code* (BRGC) [15, 16] parece estar mais perto disso que a representação binária mais tradicional, base-2. Uma ilustração disso é mostrado na Tabela 4.1.

Inteiro	Base-2	BRGC
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101

Tabela 4.1: Diferenças nas representações base-2 e BRGC de escalares inteiros

Vamos analisar daqui em diante como seria feita a conversão numa situação hipotética com os seguintes parâmetros:

- Instância bidimensional: (0,198745600; 0,443000000)
- Acurácia: 2 casas decimais $\rightarrow \lceil \log_2 10^2 \rceil = 7$ bits para representação
- Nós RAM nos discriminadores: 40

Passo-a-passo, durante o processo ocorreriam as seguintes etapas:

Estado/Ação	Representação
Original	(0,198745600; 0,443000000)
Valoração segundo acurácia	(20, 44)
Conversão para BRGC	(0001111, 0111010)
Concatenação	00011110111010

É perceptível que em nossa representação binária de uma instância alguns bits "carregam" mais informação do que outros, de acordo com a sua significância. Por exemplo, considere a comparação de 2 escalares. É mais provável que eles sejam

próximos um do outro se o bit mais significativo das suas representações BRGC coincidirem e o menos significativo não do que o contrário. Esta informação de significância dos bits seria perdida através da utilização do método padrão do WiSARD de definição dos endereços a serem acessados nos nós RAM dada uma entrada. Afinal, um mapeamento aleatoriamente estabelecido consideraria que todos os bits são igualmente importantes, o que é verdade no processamento de imagens binárias, mas não aqui.

Para evitar isso, desenvolvemos um novo algoritmo para definir esses endereços, no qual cada bit é usado para definir um número de endereços proporcional a sua significância. Apresentamos a seguir dois procedimentos: o primeiro mostra como nós definimos o mapeamento instância-para-endereços (algoritmo 1), enquanto o segundo recebe uma instância como entrada e retorna os endereços a serem usados pelos discriminadores utilizando o mapeamento previamente definido (algoritmo 2). A intuição por trás da definição do procedimento que define o mapeamento de um sistema StreamWiSARD é apresentada na sequência de passos a seguir:

1. Sejam b bits usados na representação de um valor real
2. Considere o crescimento exponencial da "importância" dos bits, como em '1, 2, ..., 2^{b-1}'
3. Possível ideal: o número de endereços em que cada bit participa igual a sua importância
4. O somatório da sequência de importância é 2^b
5. Outro possível ideal: o número de nós RAM $n = 2^b$, implicando em $f = \sqrt[b]{n} = 2$
6. Caso $n \neq 2^b$, $f \neq 2$, mas o sentido de crescimento exponencial é mantido

Considerando novamente o nosso exemplo, eis algumas informações sobre o estado final do procedimento:

- Seja $f = \sqrt[7]{40}$
- Bit menos significativo: participará de $\lfloor f^0 \rfloor = 1$ endereços
- Segundo menos significativo: $\lfloor f^1 \rfloor$ endereços
- Bit mais significativo: $\lfloor f^6 \rfloor$ endereços

Algoritmo 1: Define o mapeamento instância-para-endereços

Entrada: d, a dimensionalidade dos dados
Entrada: a, a acurácia dos valores reais
Entrada: n, o número de nós RAM em cada discriminador
Saída: map_n, um vetor de listas de referências a bits

```
b ← ⌈log_2 10^a⌉
f ← ∛n
para i ← 1 até b faça
    para j ← 1 até d faça
        Seleciona aleatoriamente ⌊f^(i−1)⌋ nós
        para cada índice dos nós selecionados, k faça
            Dado o i-ésimo bit da conversão do j-ésimo valor de uma instância
              Adiciona uma referência a esse bit a map_k
        fim
    fim
fim
```

Algoritmo 2: Mapeia uma instância para endereços

Entrada: *inst*, a instância a ser convertida
Entrada: n, o número de nós RAM em cada discriminador
Entrada: map_n, um vetor de listas de referências a bits
Saída: $addr_n$, um vetor de números inteiros, endereços de locais em nós RAM

```
para i ← 1 até n faça
    addr_i ← 0
    para cada referência (dimensão j, bit k) of map_i faça
        Seja b o k-ésimo bit da conversão-BRGC do j-ésimo valor de inst
        addr_i ← 2addr_i + b
    fim
fim
```

Diferente do mapeamento do WiSARD, o mapeamento definido para o StreamWiSARD pode formar grupos de tamanhos diferentes, no caso do número total de bits a serem usados para formar os endereços não ser divisível pelo número

de nós em cada discriminador, o que resulta na composição de discriminadores com nós RAM com endereços de tamanho diferente. Porém, a escolha aleatória de nós realizada nele acontece de modo a minimizar a diferença no tamanho do menor e do maior grupo de bits. Após a definição do mapeamento para o StreamWiSARD, caso todos os grupos de bits não tenham o mesmo tamanho, a maior diferença entre o tamanho de dois deles é unitária.

4.2 A componente *online*

Esta componente é facilmente descrita como um laço de iteração sobre os discriminadores existentes, onde a similaridade entre cada um deles e a instância sendo processada é avaliada. O primeiro discriminador encontrado cuja semelhança é considerada alta o suficiente absorve a instância.

Cada discriminador tem um limiar dinâmico de similaridade para absorção de instâncias, que cresce conforme o número de instâncias absorvidas, e assemelha-se à janela de aprendizagem do AutoWiSARD [14]: o limiar de um discriminador que já absorveu k instâncias é igual a $t = min(1, m + k/e)$, ou seja, a pontuação de similaridade mínima de uma instância a ser absorvida por este discriminador é nt, onde n é o número de nós de RAM de um discriminador. m e e são parâmetros da arquitetura, nomeados limiar de similaridade mínimo e número máximo de absorções respectivamente. Eles são a mais forte influência wsobre granularidade dos microgrupos.

Os discriminadores são mantidos em uma lista ordenada, onde o primeiro foi o último a absorver uma instância. Assim, cada vez que um discriminador aprende uma nova instância ele é movido para o início da lista. Devido à ocorrência de rajadas de casos semelhantes acreditamos que isto pode levar a um sutil ganho de desempenho.

As Figuras 4.2 e 4.3 ilustram, respectivamente, a avaliação sequencial de similaridade entre os discriminadores e a observação da entrada. Na Figura 4.2 as indicações "$A : n$" indicam que o discriminador que contém essa inscrição já absorveu, apren-

deu n observações. Vale a pena notar que a avaliação foi interrompida assim que foi obtida de um discriminador uma resposta de similaridade superior ao seu limiar, e que esta resposta não foi a melhor de todas.

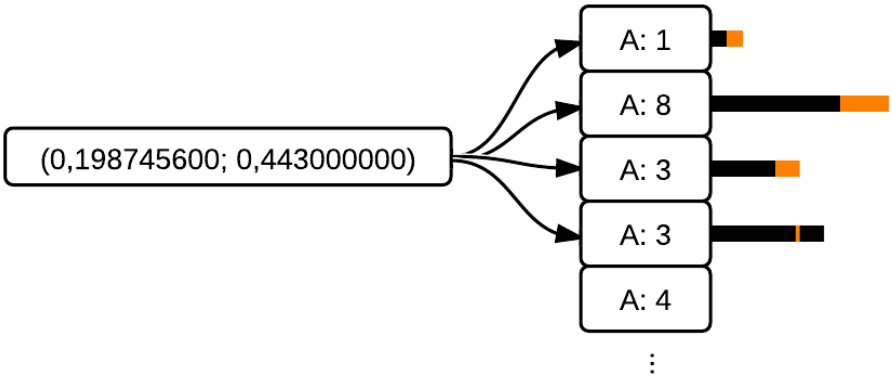


Figura 4.2: A busca por um discriminador "compatível" com a observação

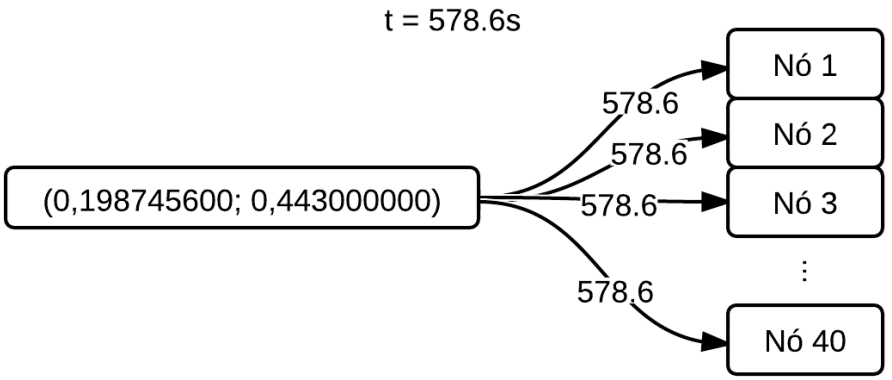


Figura 4.3: O aprendizado temporal de uma observação

Quando nenhum discriminador é considerado semelhante o suficiente para absorver uma instância, um novo é criado. É possível definir um número máximo de discriminadores a existir durante o funcionamento da arquitetura. Devido a isso, se for necessário criar espaço para a criação de um discriminador novo, dois deles considerados mais próximos um do outro são unidos. No entanto, durante os testes experimentais essa operação nunca ocorreu, devido ao pequeno tamanho do grupo de discriminadores mantido durante os mesmos. O procedimento de avaliação de similaridade de discriminadores é utilizado tanto nessa operação quanto na componente *offline*. Então, este será explicado na Seção 4.3.

É necessário explicar como neste método os dados antigos são "esquecidos". Isto é possível graças a uma outra mudança no WiSARD original: cada local nos nós

RAM guarda o último instante em que foi escrito. Com o passar do tempo, quando um valor de um endereço fica fora da janela de tempo, a posição referenciada é limpa, e quando nenhuma das posições dos nós de um discriminador está escrita, ele é descartado, pois não armazena qualquer informação. Note que isso funciona como um tipo de "*bleaching* temporal", onde as posições dos nós RAM dos discriminadores são apagadas não pela frequência de marcação, mas pelo próprio passar do tempo, sendo o limiar do *bleaching* a janela de dados adotada.

É apresentada a seguir a Figura 4.5, um fluxograma que resume a forma como a componente online da nossa arquitetura funciona.

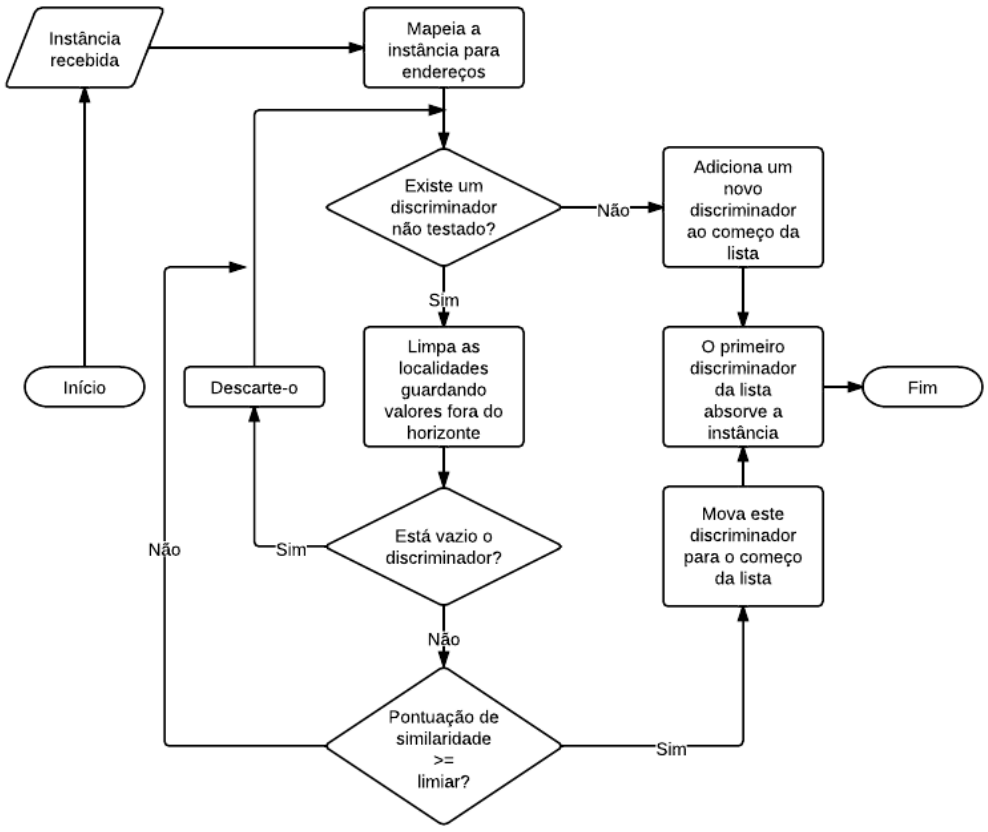


Figura 4.4: Uma representação gráfica da componente *online* do StreamWiSARD

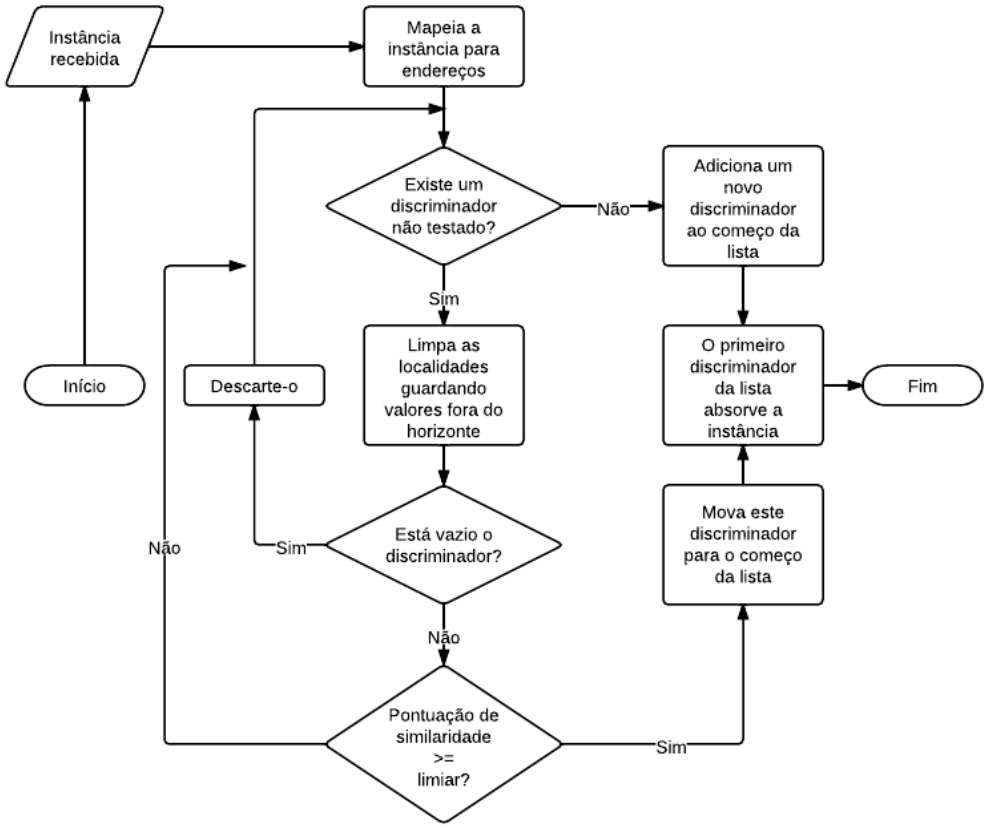


Figura 4.5: Uma representação gráfica da componente *online* do StreamWiSARD

4.3 A componente *offline*

Resumindo as ações da parte *offline* da arquitetura, esta vai agrupar discriminadores de acordo com algum critério de similaridade. É necessário definir este critério, seja qual for o método adotado para agrupamento. Um procedimento para determinar o centróide de um discriminador foi desenvolvido, de modo que a dissemelhança de um par de discriminadores foi definida como a distância entre os seus centróides.

Nós descrevemos como é determinado o centróide de um discriminador no algoritmo 3. Resumindo, o centróide é calculado de acordo com a frequência com que cada bit dos endereços armazenados nos nós RAM do discriminador vale '1'. Em outras palavras, de acordo com a moda do valor de cada bit.

Algoritmo 3: Define o centróide de um discriminator

Entrada: n, o número de nós RAM em cada discriminador
Entrada: $node_n$, o vetor de nós RAM do discriminador
Entrada: map_n, um vetor de listas de referências a bits
Entrada: d, a dimensionalidade dos dados
Entrada: a, a acurácia dos valores reais
Saída: c_d, o centróide d-dimensional do discriminador

$counter_{d,a} \leftarrow 0$
$last_{d,a} \leftarrow 0$

para $i \leftarrow 1$ **até** n **faça**
    para cada *endereço de localidade escrita em* $node_i$, *addr* **faça**
        para $j \leftarrow 1$ **até** *o tamanho de* map_i **faça**
            Seja (dimensão k, bit l) a j-ésima referência de map_i
            se *o* j*-ésimo bit de addr valer '1'* **então** $last_{k,l} \leftarrow 1$
            senão $last_{k,l} \leftarrow -1$
            $counter_{k,l} \leftarrow counter_{k,l} + last_{k,l}$
        fim
    fim
fim

para $i \leftarrow 1$ **até** d **faça**
    $brgcValue \leftarrow 0$
    para $j \leftarrow 1$ **até** a **faça**
        if $counter_{i,j} = 0$ **then** $counter_{i,j} \leftarrow counter_{i,j} + last_{i,j}$
        if $counter_{i,j} > 0$ **then** "Liga" o j-ésimo bit de $brgcValue$
    fim
    $integerValue \leftarrow convertBrgcToInteger(brgcValue)$
    $c_i \leftarrow integerValue/(10^a)$
fim

Capítulo 5

Experimentos e análise

A seguir vamos avaliar o quão bem o StreamWiSARD realiza algumas tarefas de agrupamento e discutir alguns detalhes como seus parâmetros e características singulares. Os experimentos foram realizados utilizando o MOA [17], uma vez que este fornece um ambiente de trabalho para pesquisa em agrupamento em fluxos de dados, possuindo outros algoritmos para este fim e algumas medidas de avaliação já implementados.

Foram comparados os resultados da nossa arquitetura com os obtidos pelo ClusTree [5], um dos melhores algoritmos para agrupamento de fluxo de dados conhecido, e o CluStream [3], como referência histórica para a evolução das soluções para o agrupamento de dados em fluxo contínuo. Dois tipos de conjuntos de dados foram utilizados na avaliação de desempenho: sintético, para testar a escalabilidade da arquitetura em várias dimensões, e do mundo real, como é possível que os conjuntos de dados do primeiro tipo não tenham todas as nuances destes do segundo. Nas experiências utilizando dados do mundo real apenas os microgrupos, isto é, a saída da componente *online* das alternativas foi avaliada, como o número real de grupos nesses conjuntos é desconhecido *a priori*.

A saída das alternativas foi solicitada periodicamente, em cada intervalo de tempo igual ao horizonte de dados utilizado. A qualidade destas saídas foi quantificada através das seguintes medidas: para microgrupos, pureza e entropia[18]; para (macro)grupos, precisão e revocação[19]. Todas as medidas foram feitas utilizando

os dados de entrada dentro do horizonte de tempo no instante em que a saída foi requisitada.

Considere um conjunto de conjuntos de instâncias $c = \{c_1, c_2, ..., c_{|c|}\}$, separadas segundo sua classe, um conjunto de microgrupos $m = \{m_1, m_2, ..., m_{|m|}\}$ e um conjunto de grupos $g = \{g_1, g_2, ..., g_{|g|}\}$. Sendo assim, definimos:

- **Pureza de um microgrupo** m_i, a maior porcentagem de instâncias de mesma classe de um microgrupo. Quanto maior, melhor;

$$\frac{\max_{j=1}^{|c|} |m_i \cap c_j|}{|m_i|}$$

- **Entropia de um microgrupo** m_i, um somatório da participação relativa de cada classe num microgrupo. Quanto menor, melhor;

$$-\sum_{j=1}^{|c|} \frac{|m_i \cap c_j|}{|m_i|} \times \log \frac{|m_i \cap c_j|}{|m_i|}$$

- **Precisão de um grupo** g_i, a probabilidade de uma das instâncias do microgrupo ser da classe a qual ele melhor representa. Quanto maior, melhor;

$$\frac{\max_{j=1}^{|c|} |g_i \cap c_j|}{|g_i|}$$

- **Revocação de um grupo** g_i, a probabilidade de uma das instâncias da classe a qual o microgrupo melhor representa o compor. Quanto maior, melhor;

$$\frac{\max_{j=1}^{|c|} |g_i \cap c_j|}{|c_j|}$$

- **Precisão de uma classe** c_i, a probabilidade de uma das instâncias da classe compor o microgrupo que melhor representa-a. Quanto maior, melhor;

$$\frac{\max_{j=1}^{|c|} |g_j \cap c_i|}{|g_j|}$$

- **Revocação de uma classe** c_i, a probabilidade de uma das instâncias do microgrupo que melhor representa uma classe ser daquela classe. Quanto maior, melhor.

$$\frac{\max_{j=1}^{|c|} |g_j \cap c_i|}{|c_i|}$$

Por fim, as médias dessas medidas de precisão e revocação que acabam de ser definidas são chamadas precisão e revocação dos grupos, e precisão e revocação das classes, respectivamente, e serão utilizadas adiante para avaliação global de conjuntos de grupos como g. Já os conjuntos de microgrupos como m serão avaliados pela média da pureza e média da entropia de cada um de seus elementos. Esta última média, ponderada pelo número de instâncias que cada microgrupo contém.

5.1 Dados artificiais

Através dos experimentos com datasets artificiais podemos explorar uma vasta gama de conFigurações de parâmetros para geração de conjuntos de dados com diversas características, através dos quais várias peculiaridades do StreamWiSARD podem ser identificadas. Desta feita, a seguir será avaliado o comportamento do método em diversas situações.

A não ser que explicitamente indicado os experimentos foram realizados utilizando fluxos de dados com 5 grupos reais hiperesféricos com $0,02$ unidades de raio, 25 dimensões e 10% de ruído. De cada fonte de dados foram apresentados às alternativas 100 mil instâncias, consideradas num horizonte de 1000 instâncias. Os parâmetros das mesmas foram:

- StreamWiSARD: valores reais com 1 casa decimal de acurácia, 10 nós, limiar

de similaridade mínimo 20% e número máximo de absorções 1000;

- ClusTree: Número de níveis da árvore 8, fator de ramificação 3;
- CluStream: Número máximo de microgupos 100, fator multiplicativo dos raios dos microgrupos 2.

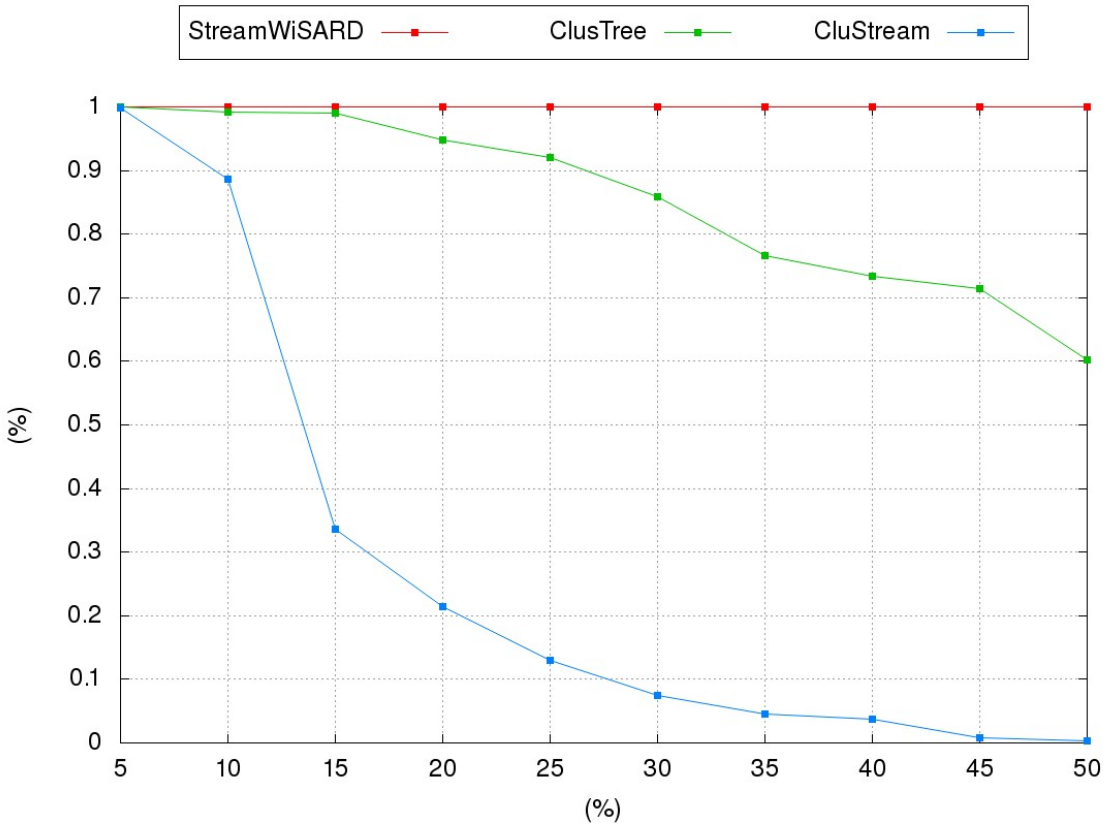
Dados com ruído. Esta é uma situação das mais frequentes, por isso a primeira abordada. A Figura 5.1 apresenta os resultados do StreamWiSARD e do ClusTree numa sequência de experimentos onde o nível de ruído adicionado aos dados importantes foi aumentado gradativamente, em passos de 5%. São mostradas apenas as medidas consideradas mais interessantes: em geral, aquelas onde houve maior disparidade entre as opções.

No quesito qualidade, o StreamWiSARD se saiu melhor que o ClusTree. Porém, é notável que conforme o aumento no nível de ruído o primeiro fica consideravelmente mais lento que o segundo. Isso é explicado pelo maior número de discriminadores que a componente *online* da arquitetura acaba criando e mantendo por certo período pelo processamento de instâncias não associadas a nenhum grupo real. Logo, fica evidente uma troca entre qualidade e velocidade nesta situação.

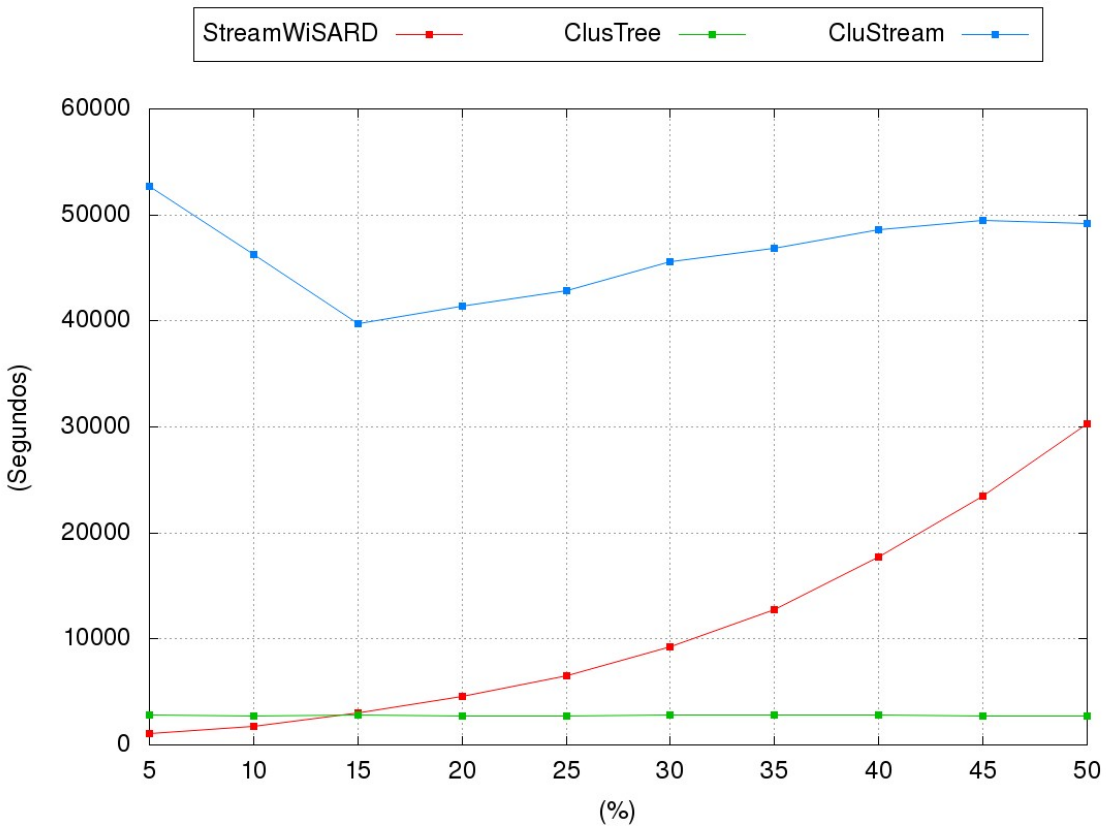
Grande número de grupos. Esta é outra situação básica, como a anterior. Sendo assim, de maneira semelhante foi feito um lote de experimentos aumentando gradativamente, em passos de 5 grupos, o número de grupos em cada execução. As similaridades não terminam por aí: os resultados são semelhantes e a justificativa é também a mesma: com o aumento no número de grupos aumenta o número microgrupos a ser mantido pela arquitetura. A Figura 5.2 apresenta os resultados.

O efeito *Hughes*. Um questionamento interessante é este: como a arquitetura se comporta lidando com dados descritos em muitas dimensões? Para que seja feita uma análise dessa situação é apresentada a seguir a Figura 5.3.

Aqui o StreamWiSARD se mostra mais rápido que o ClusTree. Não somente isso mas a taxa de crescimento do tempo de execução conforme o aumento do número de dimensões também aparenta ser menor, mediante observação do gráfico. Credita-

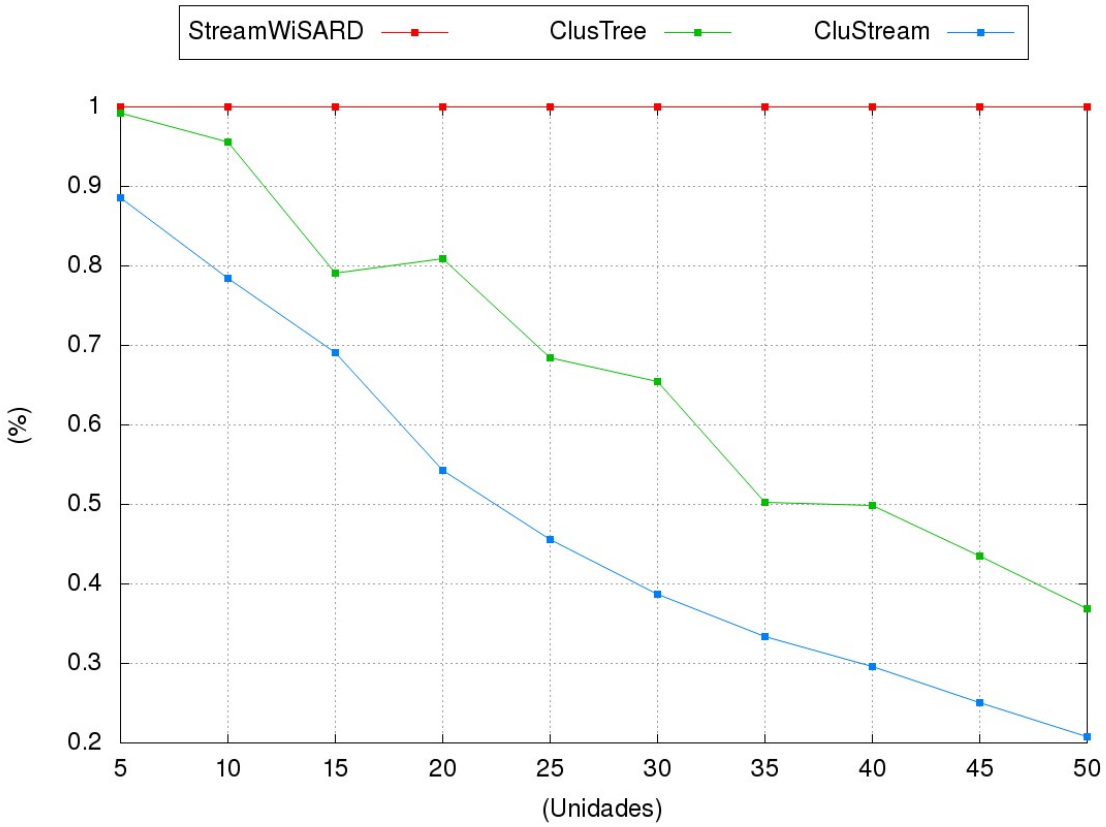


(a) Ruído × Precisão de classe

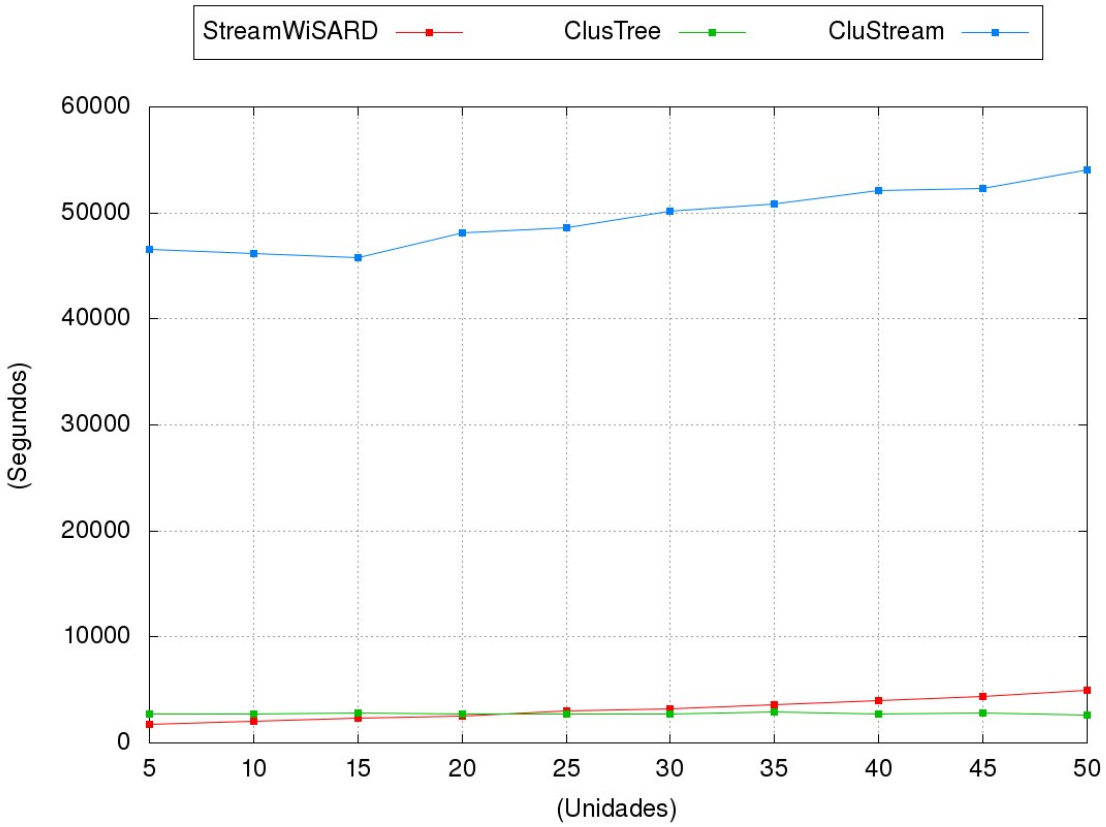


(b) Ruído × Tempo de execução

Figura 5.1: Resultados das experiências sobre ruído

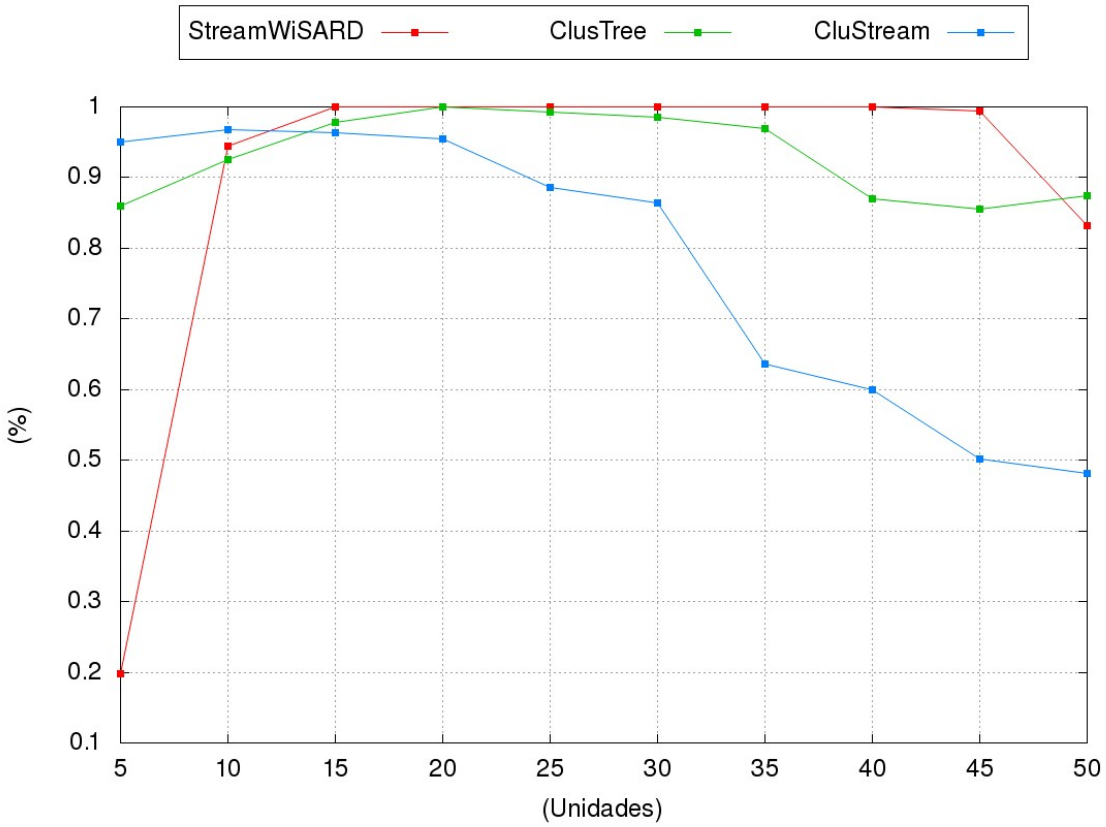


(a) Número de grupos × Precisão de classe

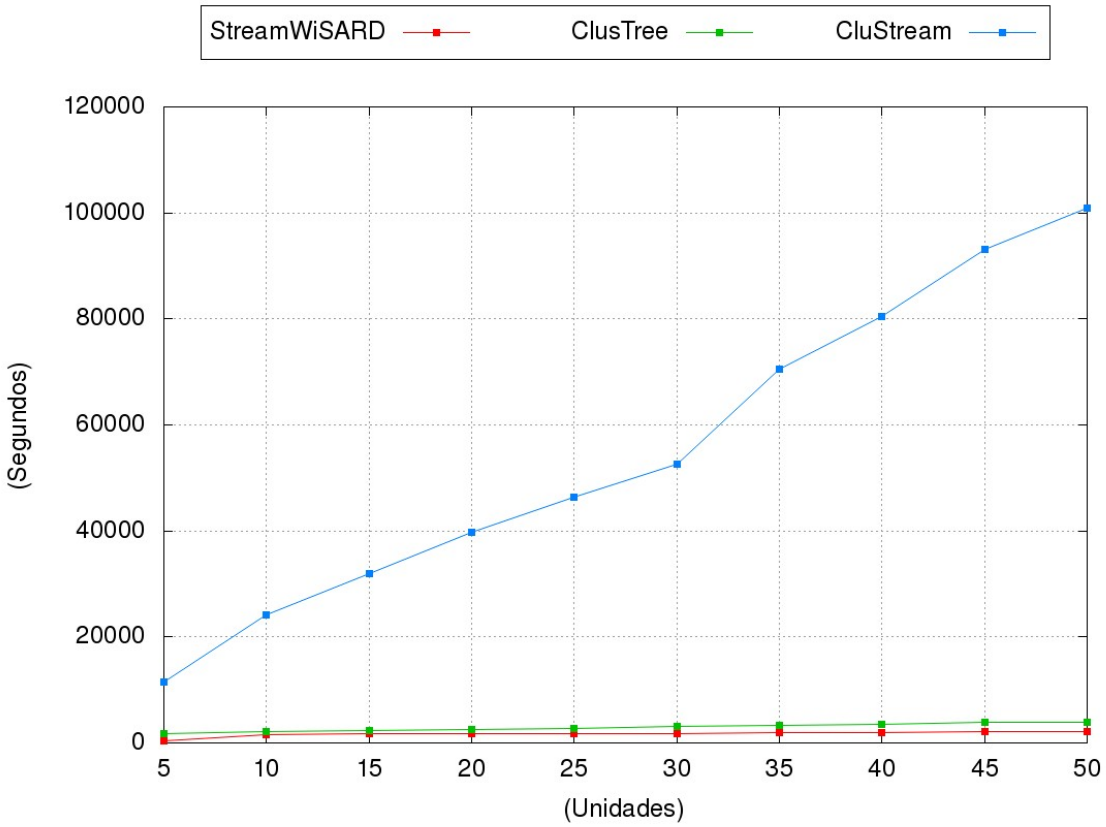


(b) Número de grupos × Tempo de execução

Figura 5.2: Resultados das experiências sobre número de grupos



(a) Dimensões × Precisão de classe



(b) Dimensões × Tempo de execução

Figura 5.3: Resultados das experiências sobre dimensionalidade

mos isso a mudança de representação feita em cada instância processada. Ilustrando isso é apresentada a Figura 5.4, um gráfico mostrando a diferença no número de bits usado para representar uma instância em sua representação original e naquela utilizada internamente pelo StreamWiSARD, considerando os parâmetro usados no experimento: valores reais com acurácia até uma casa decimal e apenas 10 nós RAM em cada discriminador. Vemos que nesse gráfico a taxa de crescimento da representação original é maior que aquela do StreamWiSARD, o que pode ser considerado o motivo para o comportamento semelhante observado na Figura 5.3b.

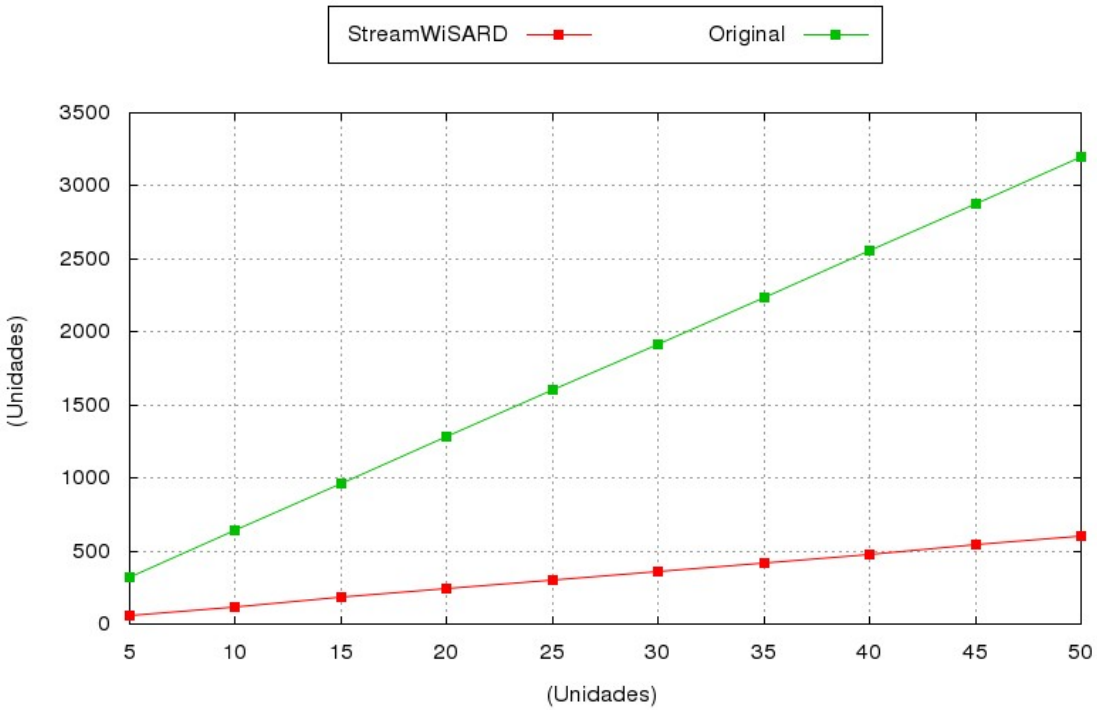


Figura 5.4: Número de bits utilizados na representação de instâncias × número de dimensões

Outro fato interessante é o valor usado como acurácia dos valores reais: apenas uma casa decimal. Isso foi assim feito pois ser esperada uma maior facilidade no agrupamento de dados conforme o aumento no número de dimensões: o nível de resolução dos dados é proporcional tanto a acurácia dos seus valores quanto a sua dimensionalidade. Cremos que isso pode explicar a consideravel queda da qualidade observada na execução utilizando apenas 10 dimensões sem que a acurácia considerada fosse ajustada.

É notável também a queda brusca na qualidade do experimento com 45 para

aquele com 50 dimensões. Assim como foi considerado interessante o aumento da acurácia assumida quando lidando com poucas dimensões, o ajuste correspondente para quando a dimensionalidade cresce é o aumento no número de nós RAM nos discriminadores, o que permite uma maior granularidade no nível de similaridade entre uma instância e um microgrupo. O fato da distância máxima entre dois pontos descritos em muitas dimensões ser maior que aquela entre dois pontos descritos em poucas pode servir ser como exemplo da utilidade do uso do maior número de nós.

Dinamismo dos grupos. Até o momento a arquitetura foi avaliado apenas em situações onde o número de grupos e as características destes eram estáticos. Mas problemas reais com tais características são a exceção, e não a regra: na prática, grupos surgem e somem, possuem densidades e tamanhos não só diferentes mas também variáveis, além de outras caratcerísticas que podem tornar a sua solução muito mais complicada. Na Tabela 5.1 são apresentados os resultados duma experiência com uma reprodução artificial desse tipo de ambiente, onde há possibilidade de variação de diversas características dos grupos. Esta experiência atesta a capacidade do StreamWiSARD de lidar mesmo com situações de alta complexidade de funcionamento.

		Grupos		Classes	
		Precisão	Revocação	Precisão	Revocação
StreamWiSARD	média	0.992	0.995	0.975	0.997
	desvpad	0.002	0.001	0.004	0.001
ClusTree	média	0.933	0.702	0.303	0.233
	desvpad	0.007	0.041	0.054	0.049
CluStream	média	0.998	0.710	0.360	0.263
	desvpad	0.001	0.036	0.036	0.037

Tabela 5.1: Resultados para o agrupamento no conjunto de alta dinamicidade

Aqui os seguintes parâmetros foram utilizados:

- Entre 5 e 55 grupos;
- A medidas dos raios varia entre 0,01 e 0,03;
- É possível tanto a divisão de 1 grupo em dois quanto o surgimento isolado de um novo grupo;

- Também é possível a união de 2 grupos em 1 e o desaparecimento isolado de um grupo.

5.2 Dados reais

O *Physiological Data Modeling Contest* (PDMC) foi um concurso realizado no ICML 2004 [20]. O conjunto de dados utilizados neste é composto dos dados temporais obtidos a partir de medidas corporais periódicas de um grupo de pessoas. Como este conjunto de dados foi originalmente usado em tarefas de classificação, todas as suas instâncias possuem um rótulo de classe segundo a atividade que estava sendo realizada quando as leituras dos sensores foram capturadas. No conjunto de dados original existe um maior número de leituras dos indivíduos identificados pelos códigos 1 e 25. Para cada um deles geramos um conjunto de dados contendo apenas suas respectivas instâncias. Nestes conjuntos de dados foram descartados todos os atributos além das leituras dos sensores (9 atributos) e o rótulo de classe. Não levamos em conta nos resultados apresentados a saída das alternativas para intervalos de tempo onde apenas uma classe estava presente, sendo nessa situação a saída trivial.

		StreamWiSARD	ClusTree	CluStream
código 1	media	0.047	0.108	0.092
	desvpad	0.043	0.093	0.091
código 25	media	0.028	0.078	0.093
	desvpad	0.021	0.065	0.082

(a) Entropia

		StreamWiSARD	ClusTree	CluStream
código 1	media	0.951	0.913	0.948
	desvpad	0.038	0.073	0.059
código 25	media	0.966	0.931	0.950
	desvpad	0.024	0.055	0.050

(b) Pureza

Tabela 5.2: Resultados para o agrupamento no conjunto de dados PDMC

Estes resultados apresentados na Tabela 5.2 são especialmente interessantes de-

vido à conFiguração de parâmetros usada: cada discriminador era composta por apenas 5 nós RAM, e cada valor real foi considerado tendo acurácia apenas até duas casas decimais. Considerando que o processo de mapeamento instância-para-endereços é uma espécie de processo de transformação de coordenadas, este não só reduziu a dimensionalidade dos dados, mas também reduziu o número de bits necessários para representar seus valores em cada dimensão: o tamanho médio dos endereços de RAM para nodos foi de 28,8 bits, contra os 64 bits geralmente usados para representar valores reais com precisão dupla.

Também foi notado que o StreamWiSARD manteve um pequeno número de microgrupos durante o processamento de todo o conjunto de dados: no máximo 19. Sendo assim, a profundidade máxima da árvore mantida pelo ClusTree foi definida para fazer com que o número de microgrupos mantidos nesta no nível das folhas fosse tão perto quanto possível deste número, para que pudéssemos verificar uma das nossas suspeitas: a utilização de uma estrutura mais poderosa para sintetizar os dados em primeira instância poderia levar a bons resultados, apesar de uma pequena quantidade de microgrupos. A Figura 5.5 apresenta como a entropia dos grupos varia durante o processamento do fluxo referente ao indivíduo 1.

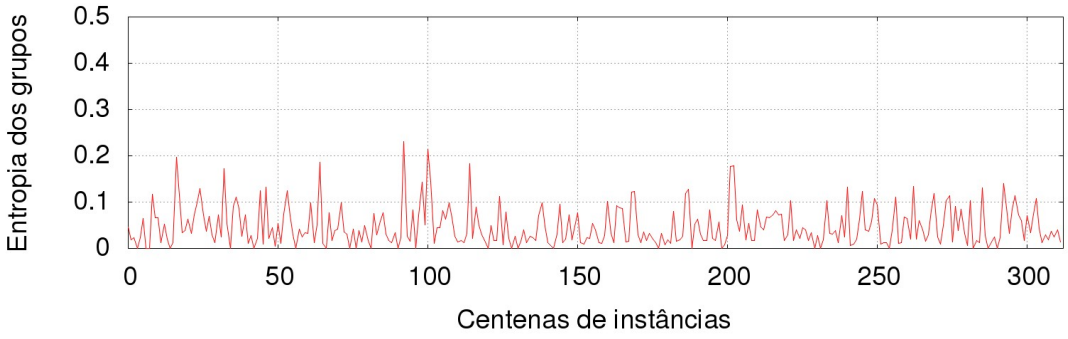


Figura 5.5: Evolução da entropia no processamento do fluxo do indivíduo 1

O conjunto de dados *Covertype* [21] também foi utilizado em nossos testes experimentais. Este foi originalmente usado em uma tarefa de classificação, mas têm sido utilizado recentemente na avaliação comparativa de várias opções para o agrupamento de fluxo de dados. Aqui foram descartados todos os atributos *booleanos*, resultando em um conjunto de dados com 10 atributos numéricos, e o rótulo de

classe. As instâncias foram processadas na mesma ordem do conjunto de dados original. Os resultados são apresentados na Tabela 5.3.

	StreamWiSARD	ClusTree	CluStream
media	0.091	0.131	0.159
desvpad	0.121	0.089	0.155

(a) Entropia

	StreamWiSARD	ClusTree	CluStream
media	0.899	0.893	0.900
desvpad	0.130	0.080	0.123

(b) Pureza

Tabela 5.3: Resultados para o agrupamento no conjunto de dados *Covertype*

Capítulo 6

Conclusão

Assim é terminada a apresentação deste trabalho. Neste foi introduzido uma nova arquitetura para agrupamento de dados em fluxo contínuo, baseada em discriminadores WiSARD. O uso de unidades básicas de agrupamento inspiradas nesse poderoso classificador se mostrou interessante, permitindo o agrupamento dos dados com qualidade ainda que com uma quantidade reduzida de microgrupos sendo mantida em todo tempo durante o processamento do fluxo. Este fato confirma uma das primeiras conjeturas que motivaram este trabalho: o uso de classificadores mais poderosos reduziria drasticamente o número de elementos necessários para resumir em primeira instância os dados apresentados.

É importante notar que certas características do WiSARD tornam o seu uso possível numa situação com restrições que inviabilizam o uso de diversos outros métodos. Aqui, a velocidade de processamento é prioritária, uma das características mais marcantes desse modelo. Outro fato importante é a capacidade de aprendizado de classes sem que haja qualquer requisito de separabilidade ou similares das mesmas. Tudo isso foi herdado pelo StreamWiSARD.

O mapeamento do StreamWiSARD, que permite a conversão de um vetor n-dimensional em um vetor de inteiros para utilização como endereços de posições a serem acessadas nos nós RAM de discriminadores, também merece destaque. É uma solução inovadora para adaptar uma entrada n-dimensional real a um sistema baseado num classificador originalmente desenvolvido para lidar com imagens binárias.

Isso nos faz refletir sobre a importância da representação dos dados em tarefas de mineração de dados. O desenvolvimento deste mapeamento representa uma considerável parcela deste trabalho, sem o qual o restante não teria sido desenvolvido. Isso serve de suporte para a seguinte consideração: em outras situações, diante de outros problemas, pode ser interessante, senão necessário, um maior investimento na especificação da representação usada do que no próprio desenvolvimento de novos métodos para solucionar o problema.

O *bleaching* segundo critérios de tempo é muito interessante. Afinal, a possibilidade de desaprender parte do conhecimento adiquirido conforme algum critério existe para um grupo seleto de ferramentas e técnicas de inteligência artificial. Na área de agrupamento de dados em fluxo contínuo há vasto espaço para uso de algoritmos e arquiteturas que permitem o esquecimento de informação gradativamente, sem o real descarte daquilo que foi aprendido mesmo há muitíssimo tempo. Verificamos isso pela grande quantidade de trabalhos desenvolvida nesse paradigma. Porém, é interessante ter disponível opções como o StreamWiSARD, que trata o esquecimento dentro de limites bem definidos.

6.1 Trabalhos futuros

Seria interessante que fossem absorvidas características de métodos como o ClusTree ou o MR-Stream, que organizam os microgrupos em diversos níveis, o que leva à redução da complexidade da operação de identificação do microgrupo mais adequado para absorver uma instância. Hierarquias de discriminadores já foram utilizadas em outras situações e talvez pudessem ser aqui também.

O mapeamento aqui apresentado merece ser observado em maiores detalhes. A questão de representação de dados na preparação da entrada de redes neurais sem pesos já foi anteriormente abordada. Uma comparação com trabalhos como o CMAC [22] seria interessante.

Em sistemas baseados no WiSARD é possível o uso de uma técnica chamada DRASiW [23], cujo nome dá uma pequena indicação do seu resultado. Através

desta é possível extrair de um discriminador WiSARD uma representação do que conhecimento aglomerado que este guarda. Talvez fosse possível a utilização dessa técnica para uma melhor definição dos centros dos microgrupos. Cardoso *et al.* [24] desenvolveram um trabalho nesse sentido, sendo porém a situação ali considerada muito mais restrita do que aquelas abordadas aqui.

Referências Bibliográficas

[1] WIKIPEDIA. "Neuron — Wikipedia, The Free Encyclopedia". 2012. Disponível em: <http://en.wikipedia.org/wiki/Neuron>. [Online; acessado em 14-Fevereiro-2012].

[2] BANDEIRA, L. C. *NC-WISARD: Uma interpretação sem pesos do modelo neural neocognitron.* Tese de Doutorado, Rio de Janeiro, RJ, Brasil, 2010.

[3] AGGARWAL, C. C., HAN, J., WANG, J., et al. "A Framework for Clustering Evolving Data Streams". In: *VLDB*, pp. 81–92, 2003.

[4] JAIN, A. K., DUBES, R. C. *Algorithms for clustering data.* Upper Saddle River, NJ, USA, Prentice-Hall, Inc., 1988. ISBN: 0-13-022278-X.

[5] KRANEN, P., ASSENT, I., BALDAUF, C., et al. "The ClusTree: indexing micro-clusters for anytime stream mining", *Knowl. Inf. Syst.*, v. 29, n. 2, pp. 249–272, 2011.

[6] ZILBERSTEIN, S. "Using Anytime Algorithms in Intelligent Systems", *AI Magazine*, v. 17, n. 3, pp. 73–83, 1996.

[7] CAO, F., ESTER, M., QIAN, W., et al. "Density-based clustering over an evolving data stream with noise". In: *In 2006 SIAM Conference on Data Mining*, pp. 328–339, 2006.

[8] ESTER, M., PETER KRIEGEL, H., S, J., et al. "A density-based algorithm for discovering clusters in large spatial databases with noise". pp. 226–231. AAAI Press, 1996.

[9] TU, L., CHEN, Y. "Stream data clustering based on grid density and attraction", *ACM Trans. Knowl. Discov. Data*, v. 3, pp. 12:1–12:27, July 2009. ISSN: 1556-4681. doi: http://doi.acm.org/10.1145/1552303.1552305. Disponível em: <http://doi.acm.org/10.1145/1552303.1552305>.

[10] WAN, L., NG, W. K., DANG, X. H., et al. "Density-based clustering of data streams at multiple resolutions", *ACM Trans. Knowl. Discov. Data*, v. 3,

pp. 14:1–14:28, July 2009. ISSN: 1556-4681. doi: http://doi.acm.org/10.1145/1552303.1552307. Disponível em: <http://doi.acm.org/10.1145/1552303.1552307>.

[11] FINKEL, R. A., BENTLEY, J. L. "Quad trees a data structure for retrieval on composite keys", *Acta Informatica*, v. 4, pp. 1–9, 1974. ISSN: 0001-5903. Disponível em: <http://dx.doi.org/10.1007/BF00288933>. 10.1007/BF00288933.

[12] ZHU, Y., SHASHA, D. "StatStream: statistical monitoring of thousands of data streams in real time". In: *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pp. 358–369. VLDB Endowment, 2002. Disponível em: <http://dl.acm.org/citation.cfm?id=1287369.1287401>.

[13] ALEKSANDER, I., GREGORIO, M. D., FRANÇA, F. M. G., et al. "A brief introduction to Weightless Neural Systems". In: *ESANN*, 2009.

[14] WICKERT, I., FRANCA, F. "Validating an unsupervised weightless perceptron". In: *Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on*, v. 2, pp. 537 – 541 vol.2, nov. 2002. doi: 10.1109/ICONIP.2002.1198114.

[15] CHAKRABORTY, U. K., JANIKOW, C. Z. "An analysis of Gray versus binary encoding in genetic search", *Inf. Sci.*, v. 156, pp. 253–269, November 2003. ISSN: 0020-0255. doi: 10.1016/S0020-0255(03)00178-6. Disponível em: <http://dl.acm.org/citation.cfm?id=957723.957729>.

[16] ROWE, J., WHITLEY, D., BARBULESCU, L., et al. "Properties of Gray and Binary Representations", *Evol. Comput.*, v. 12, pp. 47–76, March 2004. ISSN: 1063-6560. doi: http://dx.doi.org/NO_DOI. Disponível em: <http://dx.doi.org/NO_DOI>.

[17] BIFET, A., HOLMES, G., PFAHRINGER, B., et al. "MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering", *Journal of Machine Learning Research - Proceedings Track*, v. 11, pp. 44–50, 2010.

[18] LÜHR, S., LAZARESCU, M. "Incremental clustering of dynamic data streams using connectivity based representative points", *Data Knowl. Eng.*, v. 68, n. 1, pp. 1–27, 2009.

[19] MOISE, G., SANDER, J., ESTER, M. "Robust projected clustering", *Knowl. Inf. Syst.*, v. 14, pp. 273–298, March 2008. ISSN: 0219-1377. doi: 10.1007/

s10115-007-0090-6. Disponível em: <http://dl.acm.org/citation.cfm?id=1357641.1357644>.

[20] Brodley, C. E. (Ed.). *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, v. 69, *ACM International Conference Proceeding Series*, 2004. ACM.

[21] BLACKARD, J. A. *Comparison of neural networks and discriminant analysis in predicting forest cover types*. Tese de Doutorado, Fort Collins, CO, USA, 1998. AAI9921979.

[22] KOLCZ, A., ALLINSON, N. "Application of the CMAC input encoding scheme in the N-tuple approximation network", *IEE Proceedings - Computers and Digital Techniques*, v. 141, n. 3, pp. 177–183, 1994. doi: 10.1049/ip-cdt:19941004. Disponível em: <http://link.aip.org/link/?ICE/141/177/1>.

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

[24] DE O. CARDOSO, D., LIMA, P. M. V., GREGORIO, M. D., et al. "Clustering data streams with weightless neural networks". In: *ESANN*, 2011.