ARNALDO OHNO

DETECÇÃO DE MUDANÇAS EM PROBLEMAS DE CLASSIFICAÇÃO A PARTIR DE CLASSIFICADORES SOCIAIS

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito para obtenção do título de Mestre em Informática.

Curitiba

Agosto/2011

ARNALDO OHNO

DETECÇÃO DE MUDANÇAS EM PROBLEMAS DE CLASSIFICAÇÃO A PARTIR DE CLASSIFICADORES SOCIAIS

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito para obtenção do título de Mestre em Informática.

Área de Concentração: Agentes de Software

Orientador: Prof. Dr. Fabrício Enembreck

Curitiba

Agosto/2011

Pontifícia Universidade Católica do Paraná
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Informática

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DEFESA DE DISSERTAÇÃO Nº 12/2011

Aos 24 dias do mês de Agosto de 2011 realizou-se a sessão pública de Defesa da Dissertação **"Detecção de Mudanças em Problemas de Classificação a partir de Classificadores Sociais"** apresentada pelo aluno **Arnaldo Ohno**, como requisito parcial para a obtenção do título de Mestre em Informática, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Fabrício Enembreck PUCPR (Orientador)	(*assinatura*)	APROVADO (*aprov/reprov.*)
Prof. Dr. Alceu de Souza Britto Junior PUCPR		APROV.
Profª. Drª. Deborah Ribeiro Carvalho PUCPR/PPGTS		Aprov

Conforme as normas regimentais do PPGIa e da PUCPR, o trabalho apresentado foi considerado APROVADO (*aprovado/reprovado*), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora registradas no Livro de Defesas do programa.

Prof. Dr. Mauro Sérgio Pereira Fonseca
Diretor do Programa de Pós-Graduação em Informática

Programa de Pós-Graduação em Informática
PPGIa
PUCPR

AGRADECIMENTOS

À Companhia Paranaense de Energia - Copel, que propiciou as condições para a realização do mestrado.

Ao professor Fabrício Enembreck pela paciência, dedicação e disposição na orientação e, principalmente, pela confiança depositada.

Aos professores Nataniel Gomes de Oliveira, Alcides Calsavara, Heinz A. Niederheitmann Jr e Luiz Pedro Zambon pelo apoio e incentivo.

Aos amigos Regina de Cássia Nandi, Sandra Maria Zotto, Klaus de Geus e Richardison Ribeiro pelas valiosas contribuições a este trabalho.

A todas as pessoas que contribuiram de alguma forma para a realização deste projeto. Em especial aos amigos que me “suportaram”, apoiaram e incentivaram durante todo o período do mestrado.

.

Sumário

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

.

AddExp	Additive Expert ensembles
ANS	Aprendizagem Não Supervisionada
ARS	Análise de Redes Sociais
AS	Aprendizagem Supervisionada
CCS	Conjunto de Classificadores Sociais
CDC	Concept Drift Committee
DWM	Dynamic Weighted Majority
IBL	Instance Based Learning
K-NN	K-Nearest Neighbors
RNA	Redes Neurais Artificiais
SEA	Streaming Ensemble Algorithm

RESUMO

Uma das técnicas mais eficientes para manipular o problema de *concept drift* em tarefas de classificação em ambientes dinâmicos e instáveis é a técnica que utiliza conjunto de classificadores. Na literatura encontram-se várias metodologias para a construção do conjunto e a integração de seus classificadores nas decisões coletivas. Este trabalho propõe uma metodologia utilizando um modelo de reputação baseado na análise das interações entre os classificadores. Nesta abordagem, durante a fase de aprendizagem, uma rede de interação é construída utilizando os relacionamentos observados entre os classificadores. Os conceitos e métricas da teoria da análise de redes sociais aplicados nesta rede são utilizados durante a construção do conjunto e nas tarefas de decisão coletiva. Os resultados satisfatórios obtidos nos experimentos comprovam a eficiência desta abordagem em um domínio com *concept drift*.

Palavras-chave: Inteligência Artificial; Conjunto de Classificadores; *Concept Drift;* Análise de Redes Sociais.

ABSTRACT

One of the most effective techniques for dealing with the problem of concept drift in classification tasks in dynamic environments is the one that uses sets os classifiers. There are several methodologies described in the scientific literature for the construction of the ensemble and integration of its classifiers in collective decisions. This paper proposes a methodology using a reputation model based on the analysis of interactions between classifiers. In this approach, during the learning phase, an interaction network is built using the observed relationships between classifiers. Concepts and metrics of social network analysis applied in this network are used during construction of the ensemble and in the tasks of collective decision. The satisfactory results obtained in the experiments prove the effectiveness of this approach in dealing with concept drift.

Keywords: Artificial Intelligence; Ensemble Learning; Concept Drift, Social Network Analysis;

1 INTRODUÇÃO

Um dos principais assuntos estudados em Aprendizagem de Máquina é a aprendizagem de conceito. Normalmente o algoritmo de aprendizagem deve observar uma certa quantidade de exemplos e o aprendiz pode generalizar a informação fornecida por estes exemplos para outros ainda não observados. Entretanto, a aprendizagem do conceito de interesse pode depender de um contexto oculto não presente na base de treinamento inicial, como por exemplo, (i) regras de predição de tempo que variam com a estação do ano; (ii) a preferência de consumidores que pode mudar dependendo do dia da semana, da oferta de outros fornecedores, etc. Mudanças no contexto podem induzir mudanças no conceito alvo, fenômeno conhecido como c*oncept drift*. Um sistema de aprendizagem para atuar em um contexto onde ocorre *concept drift* deve ser apto a detectar tais mudanças e se adaptar rapidamente a elas.

A utilização de conjuntos de classificadores para tratar mudanças de conceitos tem sido objeto de diversas pesquisas e tem se revelado uma técnica bastante eficaz. Neste tipo de abordagem vários classificadores são treinados com um algoritmo base e suas predições individuais são combinadas para formar a predição global. Os métodos para atualização do conjunto (criação e exclusão de classificadores) e a forma de integração de seus membros para uma predição global variam de acordo com cada algoritmo. Porém, os métodos propostos na literatura são em geral limitados e empíricos: a atualização é feita de acordo com o desempenho individual dos classificadores e a integração dos membros utiliza votação simples e, numa forma mais evoluída, pesos são atribuídos ao voto de cada membror de acordo com o seu desempenho. Uma outra técnica bastante simples utiliza somente a predição do classificador com o melhor desempenho. Essas limitações acabam produzindo abordagens que são difíceis de configurar em função da quantidade envolvida de parâmetros e de generalizar para diferentes domínios.

.

1.1 Motivação e hipótese

Uma rede social é uma estrutura composta de um conjunto de atores, sendo que alguns dos atores são ligados por um conjunto de relações. A análise de rede social é estrutural, com foco nas relações entre os atores e não em seus atributos individuais. Estas relações, por sua vez, exercem influências importantes no comportamento individual.

Algumas características de conjuntos de classificadores permitem fazer uma correlação com uma rede social. No conjunto, por exemplo, as decisões são tomadas de forma colaborativa e a decisão de um classificador influencia a decisão dos demais nas próximas interações. Também é possível estabelecer uma rede de relacionamentos entre os classificadores onde a afinidade (freqüência de classificações idênticas) dos classificadores é proporcional à força do relacionamento. A partir da rede estabelecida é possível mensurar o nível de prestígio e centralidade de cada classificador utilizando as métricas da Análise de Redes Sociais. Estas métricas podem ser utilizadas para realizar inferências ("decisões sociais").

Um outro aspecto no qual a análise de rede pode ter bom resultado é na avaliação de um classificador. Na maioria dos métodos encontrada na literatura, esta avaliação é feita de acordo com o desempenho do classificador. Porém, muitas vezes ocorre que o desempenho de todos os classificadores é baixo quando analisado em toda a base de treinamento, mas quando aplicado, por exemplo, na parte estável dos dados, ele apresenta ótimo desempenho. Uma análise do classificador que não seja feita individualmente e sim por sua posição dentro da rede pode evitar que este classificador seja descartado indevidamente e seus relacionamentos atualizados de forma a maximizar a performance do conjunto como um todo.

As hipóteses anteriores pressupõem que a tomada de decisão a partir de um modelo de reputação social pode alcançar bons resultados, além de fornecer uma forma natural e elegante de tomada de decisão coletiva.

1.2 Objetivos

Este trabalho avalia a aplicação de técnicas e conceitos da análise de redes sociais para modelar o conjunto de classificadores como uma rede social. O objetivo deste trabalho é propor uma metodologia de construção de um conjunto de classificadores e de integração destes classificadores que aplique a análise de redes sociais. Nesta metodologia, a partir de uma rede de relacionamentos formada com os membros do conjunto, métricas sociais serão utilizadas para definir a reputação de cada membro dentro do conjunto. Na fase de construção do conjunto, essa reputação é utilizada para identificar classificadores que perdem sua importância e podem ser removidos. Na fase de integração, a reputação também é utilizada para definir o peso dos classificadores na decisão coletiva do conjunto. Outro objetivo deste trabalho é verificar o desempenho da metodologia proposta com um algoritmo de

aprendizagem baseado em exemplos (K-NN) e um algoritmo de aprendizagem simbólico (árvore de decisão). Um terceiro objetivo consiste em comparar o desempenho da metodologia com um outro sistema de conjunto de classificadores proposto na literatura. Para a avaliação e comparação foi utilizada uma base de dados que apresenta a característica de *concept drift* nas formas abrupta, moderada e gradual.

1.3 Estrutura do documento

Este documento está organizado da seguinte forma: no capítulo 2 é feita uma fundamentação teórica, ou seja, uma pesquisa sobre os temas envolvidos e sobre trabalhos desenvolvidos que tratam de *drift detection* utilizando conjunto de classificadores. No capítulo 3 é apresentado o algoritmo CCS, que utiliza classificadores sociais na construção e integração de classificadores de um conjunto. No capítulo 4 são apresentados os resultados obtidos e uma comparação da metodologia proposta com outro método encontrado na literatura. No capítulo 5 é apresentada a conclusão.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo faz um resumo dos temas envolvidos na pesquisa, bem como alguns trabalhos realizados sobre estes temas. Na primeira seção são apresentados conceitos e definições de Aprendizagem de Máquina Supervisionada; a seguir, são apresentados os métodos de classificação que utilizam a técnica de conjunto de classificadores; na terceira seção são apresentados o problema de *Drift Detection* e alguns algoritmos que utilizam conjuntos de classificadores para lidar com este problema; na última seção é apresentada a análise de redes sociais.

2.1 Aprendizagem de Máquina

Aprendizagem de Máquina é a área de Inteligência Artificial cujo objetivo é o desenvolvimento de técnicas computacionais de aprendizado e a construção de sistemas capazes de adquirir conhecimento de forma automática.

Segundo Mitchell (1997), "pode ser dito que um programa computacional aprendeu com uma experiência *E*, no que diz respeito a alguma classe de tarefas *T* e uma medida de performance *P* se sua performance *P* nas tarefas do tipo *T* melhorou com a experiência *E*".

O aprendizado utiliza o princípio da indução com o intuito de obter conclusões genéricas a partir de um conjunto de exemplos. O aprendizado indutivo pode ser dividido em Aprendizado Supervisionado (AS) e Aprendizado Não-Supervisionado (ANS). O AS é utilizado para classificação dos exemplos em classes predefinidas, enquanto o ANS é utilizado para agrupamentos de exemplos semelhantes. O tipo de aprendizado de interesse para a pesquisa deste trabalho é o supervisionado.

Para Russel e Norvig (2004), a tarefa da inferência indutiva no problema de aprendizagem supervisionada envolve a aprendizagem de uma função[1] a partir de um conjunto de exemplos de suas entradas e saídas, denominado conjunto de treinamento. Formalmente, um exemplo é representado por um par *(x,f(x))*, onde *x* é a entrada e *f(x)* é a saída da função *f* aplicada a *x*. O que torna a aprendizagem difícil é que, na verdade, a função encontrada é uma aproximação da função *f*, que é chamada de hipótese (*h*). O termo "hipótese" vem do fato de que o resultado da aplicação de uma inferência indutiva não

[1] Kohavi (1995) define um classificador como a função que mapeia um exemplo não classificado para uma classe, utilizando estruturas de dados internas.

preserva, necessariamente, a verdade, mesmo que o conjunto de premissas utilizado na inferência seja verdadeiro. Embora não seja possível garantir a validade da hipótese, pode-se atribuir a ela um certo grau de confiança. Dizemos que uma hipótese é boa se generalizar bem, ou seja, se classificar corretamente exemplos ainda não vistos. De acordo com Russel e Norvig (2004), a indução do conceito pode ser vista como uma busca, num espaço pré-definido de hipóteses, pela hipótese que melhor classifica os exemplos de treinamento. Tal hipótese vai representar a descrição do conceito.

Uma forma simples de entender o processo de modelagem de uma função é através da aprendizagem de conceito, uma vez que a maior parte do processo de aprendizagem envolve a aquisição de conceitos gerais a partir de exemplos de treinamento específicos. Segundo Mitchell (1997) as pessoas, por exemplo, estão continuamente aprendendo conceitos como pássaro, carro, "situações que devo estudar mais para passar no exame", etc. Cada conceito pode ser visto como um subconjunto de objetos ou eventos de um conjunto maior (ex.: pássaros podem ser vistos como subconjunto de animais). Um conceito também pode ser visto como uma função booleana definida sobre este conjunto maior (ex.: uma função definida sobre animais, cujo valor é *True* para pássaros e *False* para outros animais).

O exemplo a seguir foi utilizado por Mitchell (1997) para ilustrar o processo de aprendizagem de conceito: Seja a tarefa de aprender o conceito "*dias que John pratica seu esporte náutico preferido*". A Tabela 1 apresenta um conjunto de dias, cada dia representado por um conjunto de atributos. O atributo alvo (ou conceito alvo) é o atributo que indica se John pratica ou não o esporte. A tarefa consiste em aprender a predizer se John pratica o esporte para um dia qualquer, arbitrário, baseando-se nos valores dos outros atributos. Para isto, deve ser encontrada uma relação entre os valores dos atributos e o valor do conceito alvo.

Clima	Temperatura	Umidade	Vento	Água	Previsão	PraticaEsporte
Ensolarado	Quente	Normal	Forte	Quente	Mesma	Sim
Ensolarado	Quente	Alta	Forte	Quente	Mesma	Sim
Chuvoso	Frio	Alta	Forte	Quente	Mudança	Não
Ensolarado	Quente	Alta	Forte	Fria	Mudança	Sim

Tabela 1 - Exemplos para o conceito alvo PraticaEsporte (Mitchell, 1997)

Representando uma hipótese na forma de um vetor de restrições especificando os valores dos atributos, o formato da hipótese seria:

<Clima, Temperatura, Umidade, Vento, Água, Previsão>

Os valores possíveis para um atributo são: “?” (indicando que qualquer valor é aceitável para o atributo); “0” (indicando que nenhum valor é aceitável para o atributo); um valor válido dentre os valores possíveis para o atributo (ex. Chuvoso para o atributo Clima). Para ilustrar que “John pratica seu esporte náutico preferido somente em dias frios e com alta umidade” a seguinte descrição seria suficiente:

< ?, Frio , Alta , ? , ? , ? >

Quando um exemplo *x* satisfaz todas as restrições da hipótese *h*, então *h* classifica *x* como um exemplo positivo: *h(x)* = 1 (na Tabela 1, o exemplo 3 é classificado como positivo para a hipótese anterior). Por outro lado, se o exemplo x não satisfaz todas as restrições de *h*, então *x* é classificado como exemplo negativo: *h(x)* = 0.

2.1.1 Classificação dos modelos de aprendizagem

No exemplo utilizado anteriormente, a hipótese (ou modelo) foi ilustrada como um vetor de restrições, mas o processo é o mesmo para qualquer tipo de modelo como, por exemplo, uma árvore de decisão ou uma rede neural. De acordo com os algoritmos de aprendizagem utilizados para obtenção do modelo podem ser definidos os paradigmas de Aprendizagem de Máquina:

1. **Simbólico**: os algoritmos buscam aprender construindo representações simbólicas de um conceito por meio da análise de exemplos e contra-exemplos do conceito, como expressão lógica, regras ou rede semântica e a árvore de decisão.

A árvore de decisão (Witten e Frank, 2005) é um dos métodos mais conhecidos e utilizados para inferência indutiva. Árvores de decisão são tolerantes a ruídos (exemplos com os mesmos valores de atributos, porém com diferentes classes) e utilizadas em diagnóstico de imagens médicas e análise de risco de crédito, por exemplo. A maioria dos algoritmos baseados em árvores de decisão utiliza o método dividir para conquistar (*divide-and-conquer method*), método em que o problema é sucessivamente dividido até que a solução possa ser encontrada. Seguindo este método, os classificadores baseados em árvores de decisão procuram encontrar formas de dividir sucessivamente o universo de exemplos de treinamento em vários subconjuntos, criando para tal nós contendo as condições que caracterizam esses exemplos, até que cada um deles contemple apenas uma classe ou até que uma das classes demonstre uma clara maioria não justificando posteriores divisões, produzindo nessa situação uma folha contendo a classe majoritária. O exemplo ilustrativo a seguir foi baseado em (Quinlan, 1993).

Clima	Temperatura(F)	Umidade (%)	Vento	Classe
Ensolarado	75	70	Forte	Joga
Ensolarado	80	90	Forte	Não joga
Ensolarado	85	85	Fraco	Não joga
Ensolarado	72	95	Fraco	Não joga
Ensolarado	69	70	Fraco	Joga
Nublado	72	90	Forte	Joga
Nublado	83	78	Fraco	Joga
Nublado	64	65	Forte	Joga
Nublado	81	75	Fraco	Joga
Chuvoso	71	80	Forte	Não joga
Chuvoso	65	70	Forte	Não joga
Chuvoso	75	80	Fraco	Joga
Chuvoso	68	80	Fraco	Joga
Chuvoso	70	96	Fraco	Joga

Tabela 2 – Conjunto de treinamento (Quinlan, 1993)

Dado o conjunto de treinamento (Tabela 2) com quatro atributos mais um constituindo o atributo classe com dois valores de domínio, selecionando o atributo Clima e aplicando o método de divisão e conquista, o conjunto será dividido em três subconjuntos: um subconjunto para clima Ensolarado, outro para clima Nublado e outro para clima Chuvoso. O subgrupo para clima Nublado possui somente exemplos da classe Joga e, portanto não precisa mais ser subdividido. Entretanto, os outros dois subconjuntos possuem as duas classes e devem ser divididos novamente. Se o subconjunto com clima Ensolarado é dividido em subgrupos com umidade $\leq 75\%$ e $> 75\%$ e a subconjunto com clima Chuvoso é dividido em subconjuntos com vento Forte e Fraco, todos os subconjuntos vão conter somente uma classe. Os subconjuntos finais estão representados na. A árvore de decisão correspondente está representada na Figura 1.

Basicamente, a construção da árvore ocorre da seguinte forma: um nó descendente do nó raiz é criado para cada valor possível do atributo do nó raiz. Os exemplos de treinamento são classificados de acordo com o valor que possuem para este atributo e associados com o correspondente nó. O processo é repetido em cada nó descendente, com os exemplos de treinamento associados com este nó, selecionando o melhor atributo para ser testado naquele ponto da árvore.

Clima	Temperatura(F)	Umidade (%)	Vento	Classe
Ensolarado	75	70 (≤70)	Forte	Joga
Ensolarado	69	70 (≤70)	Fraco	Joga
Ensolarado	80	90 (>70)	Forte	Não joga
Ensolarado	85	85 (>70)	Fraco	Não joga
Ensolarado	72	95 (>70)	Fraco	Não joga
Nublado	72	90	Forte	Joga
Nublado	83	78	Fraco	Joga
Nublado	64	65	Forte	Joga
Nubl do	81	75	Fraco	Joga
Chuvoso	71	80	Forte	Não joga
Chuvoso	65	70	Forte	Não joga
Chuvoso	75	80	Fraco	Joga
Chuvoso	68	80	Fraco	Joga
Chuvoso	70	96	Fraco	Joga

Tabela 3 – Subconjuntos resultantes do método Dividir para Conquistar (Quinlan, 1993)

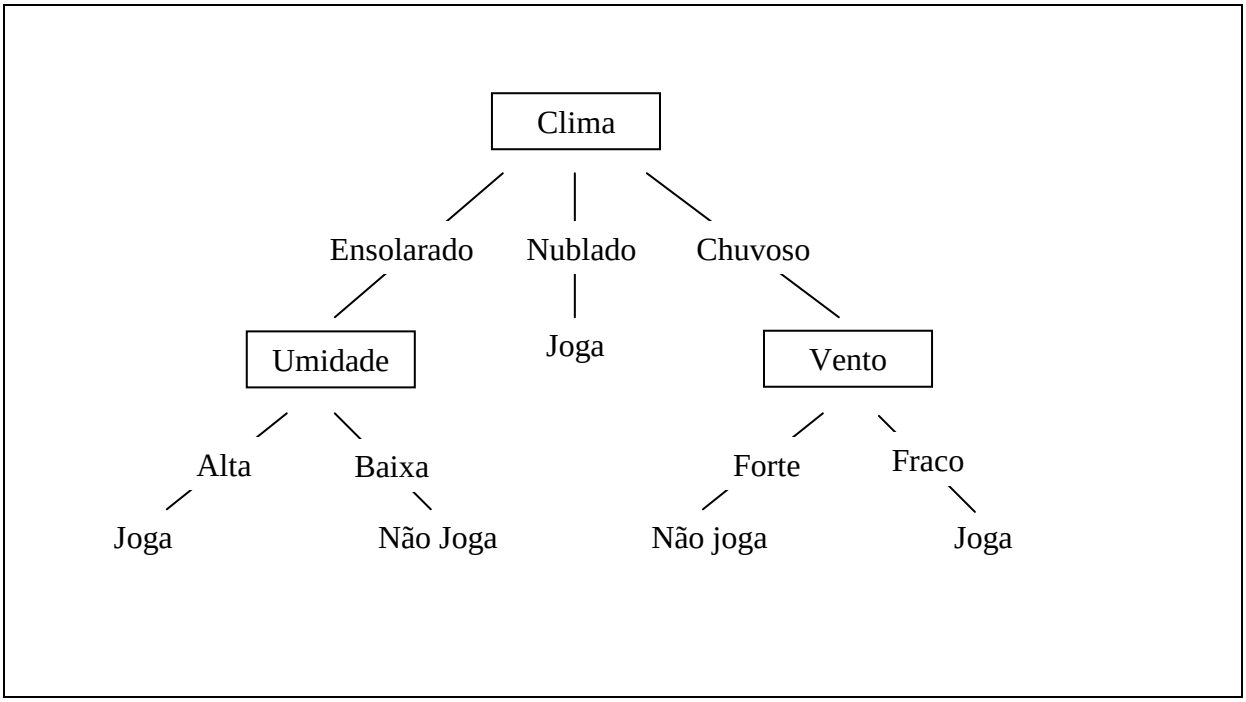


Figura 1 – Árvore de decisão para o conceito alvo Jogar Tênis (Mitchell, 1997)

O procedimento descrito é a base da maior parte dos algoritmos para aprendizagem de árvores de decisão como o ID3 (*Iterative Dichotomiser 3*), desenvolvido por Quinlan (1986). A escolha central no algoritmo ID3 é a seleção do atributo a ser testado em cada nó da árvore. Para identificar o atributo que melhor separa os exemplos de treinamento de acordo com o conceito alvo, o ID3 utiliza uma propriedade estatística chamada ganho de informação. Para entender esta propriedade, é necessário entender um conceito chamado entropia: na teoria da informação, quanto menos informações sobre um sistema, quanto maior a quantidade possível de arranjos, maior a entropia[2]. Numa coleção de exemplos *S*, contendo exemplos positivos e negativos de um conceito alvo, a entropia de *S* relativa a este conceito é dada pela Fórmula 1 onde *p+* e *p-* são, respectivamente, as proporções de exemplos positivos e negativos em S. Note que se existem somente exemplos positivos, $p+=1$ e a entropia neste caso será zero.

$$Entropia(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Fórmula 1 – Entropia de uma coleção de exemplos S relativa a um conceito alvo booleano

O ganho de informação é uma medida que indica a redução esperada na entropia causada pela partição dos exemplos de acordo com um determinado atributo. A Fórmula 2 calcula o ganho da informação (ou informação fornecida sobre o conceito alvo) sobre o conceito alvo de uma coleção *S* dado o valor de um atributo *A*. É esta medida que o ID3 utiliza para selecionar o melhor atributo em cada passo de construção da árvore.

$$Ganho(S, A) = Entropia(S) - \sum_{v \in \{A\}} \frac{|S_v|}{|S|} Entropia(S_v)$$

Fórmula 2 – Ganho de informação

Quinlan (1993) desenvolveu o algoritmo C4.5, que é uma extensão do ID3. Algumas das evoluções do C4.5 em relação ao ID3 são: (i) manipulação de atributos discretos e contínuos; (ii) manipulação de dados de treinamento com valores de atributos desconhecidos e ruído; (iii) métodos para geração de regras a partir de árvores; (iv) método de poda da árvore: uma vez criado, remove os galhos desnecessários substituindo-os por nós folhas. Mais recentemente, Quinlan apresentou a versão C5.0/See5 como evolução do C4.5.

[2] Na teoria da informação, a quantidade de informação de uma mensagem é definida como o menor número de bits necessários para conter todos os valores e significados desta mensagem. Portanto, a quantidade da informação de uma mensagem é definida pela entropia da mensagem.

2. **Estatístico**: os algoritmos utilizam modelos estatísticos para representar o conceito em questão. Dentre os métodos estatísticos, destacam-se os de aprendizado Bayesiano, onde cada exemplo de treinamento pode aumentar ou diminuir a probabilidade estimada de que uma hipótese está correta. O leitor pode encontrar maiores detalhes desse modelo de aprendizagem em (Witten e Frank, 2005) e (Russel e Norvig, 2004), uma vez que este modelo de aprendizagem não faz parte do escopo deste trabalho.

3. **Baseado em exemplos (*Instance based*)**: os algoritmos classificam novos exemplos por meio de exemplos similares conhecidos. Este método simplesmente armazena os exemplos de treinamento. Quando um novo exemplo de teste deve ser classificado, um conjunto de exemplos similares é buscado na memória e utilizado para classificar o novo exemplo. Este método assume que os exemplos podem ser representados como pontos no espaço Euclidiano e cada atributo corresponde a uma dimensão. A similaridade é medida de acordo com a distância entre o novo exemplo e os exemplos armazenados. A menor distância indica maior similaridade. O algoritmo mais elementar é o k-NN (*k-Nearest Neighbors*) (Aha, 1991), que utiliza os vizinhos (exemplos) mais próximos (em termos da distância euclidiana) para selecionar a classe mais frequente entre estes vizinhos. Ao contrário dos demais tipos de aprendizado apresentados, onde os dados de treinamento são generalizados antes de uma consulta (e por isto chamados de algoritmos ansiosos), no aprendizado baseado em exemplos a generalização é feita somente no momento da classificação de novos exemplos. Por isto, os algoritmos desse paradigma recebem o apelido de preguiçosos[3]. Nesta categoria estão os algoritmos da família IBL(*Instance Based Learning*) (Aha, et al., 1991), da qual se destaca o IB3 (Aha, et al., 1991). O resultado dos algoritmos IBL é uma função que faz o mapeamento de exemplos para classes, chamada descrição de conceito ou simplesmente conceito. A descrição de conceito inclui um conjunto de exemplos armazenados e informação sobre o desempenho destes exemplos durante a tarefa de classificação (quantidade correta e incorreta de predições). O *framework* que descreve todos os algoritmos IBL possui três componentes: (1) Função de similaridade, que calcula a similaridade entre um exemplo de treinamento e os exemplos que participam do treinamento; (2) Função de classificação, que fornece a classificação de um exemplo. Para isto utiliza o resultado da função de similaridade e os registros de desempenho de classificações anteriores dos exemplos que descrevem o conceito; (3) Atualizador da descrição de conceito, que é o responsável pelos registros de desempenho

[3] Enquanto os algoritmos preguiçosos *(lazy learning algorithms*) dispendem menor custo computacional do que os ansiosos (*eager learning algorithms*) durante a fase de treinamento, eles requerem maior capacidade de armazenamento e maior custo computacional na fase de classificação.

de classificação e pela decisão de qual exemplo incluir na descrição do conceito. Recebe o exemplo, o resultado da função de classificação e a descrição atual do conceito. É o responsável pela modificação da descrição do conceito.

Aha (1991) apresenta o algoritmo IB3 como uma evolução dos algoritmos IB1 e IB2: o algoritmo IB1 corresponde ao algoritmo do vizinho mais próximo (k-NN) padrão. Os algoritmos de aprendizagem baseados em exemplos normalmente requerem muitos recursos de memória e tempo de processamento no processo de classificação, uma vez que o exemplo a ser classificado deve ser comparado com todos os exemplos disponíveis; o IB2 já é uma versão que possui um critério para armazenamento de um novo exemplo. Quando um novo exemplo é classificado, ele somente é armazenado na descrição do conceito se sua classificação for incorreta; caso contrário, o exemplo é descartado. Mas apesar de economizar recursos de memória e processamento, o IB2 é sensível a ruído nos dados; finalmente, o IB3 é uma versão mais robusta do IB2: ele mantém um registro de classificação com cada exemplo armazenado. Este registro quantifica a taxa de classificação de um exemplo, ou seja, a taxa de precisão de um exemplo quando ele é utilizado para classificar um novo exemplo. A informação é utilizada para estimar a precisão futura do exemplo. O algoritmo também usa um teste de significância para determinar se um exemplo pode ser relevante para futuras classificações ou trata-se de ruído. Se tratar-se de ruído, é descartado. O algoritmo armazena um exemplo se a sua precisão é significativamente maior que a frequência observada de sua classe e remove-o se a sua precisão é significativamente menor.

Representado na Figura 2, o algoritmo IB3 mantém um registro de classificação (isto é, o número de tentativas de classificação corretas e incorretas) com cada exemplo salvo. Um registro de classificação resume o desempenho de classificação de um exemplo sobre exemplos de treinamento que são apresentados na sequência. IB3 emprega um teste de significância para determinar quais exemplos são bons classificadores e quais são ruído. Os bons são usados para classificar exemplos apresentados na sequência e os demais são eliminados da descrição do conceito.

4. **Conexionista**: este paradigma é representado pelas redes neurais artificiais (RNA). Inspirado no modelo biológico do sistema nervoso cujos sistemas de aprendizagem são construídos de redes complexas de neurônios interconectados. De forma análoga, as RNA são construídas a partir de um conjunto de simples unidades interconectadas, onde cada unidade recebe entradas (possivelmente as saídas de outras unidades) e produz uma simples saída (que pode se tornar entrada de outras unidades) (Carling, 1992). Outras referências sobre este modelo de aprendizagem são (Haykin,1994) e (Mitchell, 1997).

5. **Evolutivo (Genético)**: neste paradigma, os algoritmos apresentam uma analogia direta com a teoria da Evolução das Espécies de Darwin, que indica que os mais bem adaptados ao ambiente sobrevivem. Um classificador evolutivo (ou genético) (Mitchell, 1997) consiste de uma população de elementos de classificação que competem para fazer a predição: os elementos com um desempenho fraco são descartados. Elementos mais fortes proliferam, produzindo variações de si mesmos.

Algoritmos como árvores de decisão, RNA e Naïve Bayes são denominados algoritmos instáveis, pois podem gerar modelos muito diferentes quando aplicados a conjuntos de dados com pequenas diferenças entre si, tornando limitada a sua utilização em ambientes ruidosos ou dinâmicos.

$DC \leftarrow \varnothing$ (DC = *Descrição do conceito*)

Para cada $x \in$ *Conjunto de treinamento faça*

1. *Para cada* $y \in DC$ *faça*
   $Sim[y] \leftarrow Similaridade(x, y)$
2. *Se* $\exists \{y \in DC \mid aceitavel(y)\}$
   então $y_{max} \leftarrow y \in DC$ *com máximo* $Sim[y]$
   senão
   2.1. $i \leftarrow$ *um valor em* $[1, |DC|]$
   2.2 $y_{max} \leftarrow y \in DC$ *que é o* $i-$*ésimo exemplo mais similar a* x
3. *Se* $classe(x) \neq classe(y_{max})$
   então classificação $\leftarrow$ *correta*
   senão
   3.1. *classificação* $\leftarrow$ *incorreta*
   3.2. $CD \leftarrow CD \cup \{x\}$
4. *Para cada* y *em* CD *faça*
   Se $Sim[y] \geq Sim[y_{max}]$
   então
   4.1. *Atualiza registros de classificação de* y
   4.2. *Se registro de* y *é significativamente pobre*
   então $CD \leftarrow C - \{y\}$

Figura 2 – Algoritmo IB3 baseado em (Aha, et al., 1991)

2.2 Métodos de classificação utilizando conjunto de classificadores

Como apresentado anteriormente, os algoritmos tradicionais de aprendizagem supervisionada geram um único modelo, que será utilizado para classificar novos exemplos. Os métodos de classificação que utilizam um conjunto (ou comitê) de classificadores surgiram da ideia de construir um conjunto de classificadores a partir dos dados de treinamento, de forma que a predição da classe de um exemplo desconhecido fosse feita através da combinação das predições feitas pelos classificadores deste conjunto. Estes métodos foram desenvolvidos para tentar obter um melhor desempenho do que classificadores individuais. A ideia geral destes algoritmos é executar várias vezes o algoritmo de aprendizagem (o algoritmo base de aprendizagem) e combinar as hipóteses resultantes em uma hipótese conjunta (por exemplo, via voto majoritário) e pode ser resumida da seguinte forma:

1. Criar múltiplas amostras de dados de treinamento
2. Construir múltiplos classificadores (um para cada amostra)
3. Combinar os classificadores para obter a classificação global

Os métodos mais populares de aprendizagem por conjunto são *Bagging, Boosting e Random Subspace*. Todos envolvem manipulação de dados de treinamento para gerar diferentes classificadores, que serão combinados em um classificador composto, cuja classificação será obtida através da votação dos classificadores. Muitas pesquisas mostram que métodos baseados em conjuntos de classificadores geralmente levam a melhoras significativas em uma série de problemas de aprendizagem: (Breiman, 1996) apresenta estudos onde o método *bagging* é aplicado em árvores de decisão sobre dados reais e simulados e o resultado é a diminuição de erros de classificação e (Quinlan, 1996) mostra melhora na precisão de predição quando as técnicas de *boosting* e *bagging* são aplicadas em um sistema de aprendizagem de árvores de decisão. Freund e Schapire (1996) comparam a performance da aplicação dos dois métodos em uma série de base de estudos onde os resultados indicam melhor desempenho de *boosting*. De acordo com Skurichina e Duin (2002), as pesquisas realizadas com *boosting, bagging* e *Random Subspace* mostram que *Random Subspace* pode superar *boosting* e *bagging*. Nas seções seguintes estes métodos são detalhados.

2.2.1 Bagging (Boostrap aggregating)

Bagging é um algoritmo de aprendizagem por conjunto introduzido por (Breiman, 1996) para melhorar a aprendizagem em termos de estabilidade e precisão na classificação. De maneira geral, o método gera várias amostras *bootstrap*[4] do conjunto de treinamento e para cada amostra gera um classificador. A predição para um novo exemplo é feita pela votação dos classificadores individuais, sendo que cada classificador tem o mesmo peso na votação. Os resultados da pesquisa em (Breiman, 1996) mostram que esta técnica apresenta bons resultados quando o algoritmo de aprendizado utilizado é instável. Este algoritmo, apresentado na Figura 3, consiste dos seguintes passos:

Entrada : *conjunto de treinamento S*, *Indutor L*, *nr de amostras bootstrap T*

1. *Faça* $i = 1$ *até* T {
2. $S' =$ *amostra bootstrap de S* (*amostra com reposição*)
3. $Ci = L(S')$
4. }
5. $C^*(x) = \underset{y \in Y}{\arg\max} \sum_{i:C_i(x)=y} 1$ (*a classe y mais predita*)

Saída : *classificador* C^*

Figura 3 - Algoritmo *Bagging*, baseado em (Bauer and Kohavi,1999)

1. Cria *T* amostras *bootstrap* (geradas através de amostragem de *m* exemplos do conjunto de treinamento, com reposição, ou seja, os exemplos podem estar em várias amostras): $S_1, S_2, ..., S_T$;

2. Um classificador C_i é construído sobre cada amostra B_i ;

3. Um classificador final C^* é construído de cada amostra B_i, cuja classe predita é a classe predita mais vezes pelos classificadores C_i

[4] Técnica estatística onde o conjunto de dados é reamostrado em várias amostras, através de sorteio aleatório e com reposição.

2.2.2 Boosting

Boosting é um método introduzido por Freund e Schapire (1995) para melhorar a performance de um classificador fraco (um classificador de baixo desempenho). Baseado na observação de que encontrar várias regras fracas é mais fácil que encontrar uma única regra de alta precisão, *boosting* se refere a um método geral para produzir regras de predição de muita precisão através da combinação de regras fracas e pouco precisas. Neste trabalho, Schapire utiliza como exemplo um filtro de email que possa distinguir *spam*. A abordagem poderia ser: selecionar exemplos de *emails* dos tipos *spam* e não-*spam* que servirão de entrada para um algoritmo de aprendizagem de máquina cujo objetivo é produzir uma classificação ou uma regra de predição. Dado um novo *email*, tal regra tenta predizer se é um *spam* ou não. O objetivo é gerar uma regra que faça a predição mais precisa possível. Mas regras pouco precisas como, por exemplo: "Se a frase 'compre agora' ocorre no *email*, então é um *spam*" dificilmente cobrirão todas as mensagens de *spam*. A ideia do algoritmo *boosting* é gerar várias regras fracas como estas e combiná-las de modo a formar uma regra mais forte que qualquer uma delas.

Neste método, parte-se de um conjunto de dados inicial onde todos os exemplos possuem um peso uniforme e gera-se um classificador utilizando o peso dos exemplos como probabilidade de escolha para o conjunto de treinamento. Cria-se novo conjunto de dados de acordo com o resultado do classificador: os exemplos classificados corretamente pelo classificador têm seu peso reduzido e os exemplos classificados incorretamente têm seu peso aumentado. O algoritmo é sensível aos pesos dos exemplos contidos no conjunto de dados, de forma que é “forçado” a classificar corretamente os exemplos com maior peso, pois a próxima iteração utilizará o conjunto de dados com peso modificado para gerar o novo classificador. Esse processo é repetido um certo número de iterações, gerando novos classificadores. O classificador final é formado pela votação dos classificadores individuais, porém o voto de cada classificador possui um peso que está diretamente relacionado ao desempenho individual do classificador. Percebe-se que essa técnica também gera classificadores que são complementares entre si, pois cada classificador tende a não cometer erros nos exemplos classificados incorretamente por outros classificadores, o que faz com que essa técnica também seja eficiente para algoritmos instáveis.

Os algoritmos de *Boosting* diferem principalmente na forma como são distribuídos pesos aos exemplos e na forma de como as hipóteses são combinadas. A Figura 4 apresenta o

algoritmo *Adaboost* , introduzido em (Freund e Schapire, 1996), o mais utilizado em estudos teóricos e testes empíricos. Este algoritmo consiste dos seguintes passos:

1. Cria T amostras de treinamento com pesos: S_1, S_2 , ..., S_T . Estas amostras são geradas sequencialmente

2. Um classificador C_i é construído sobre cada amostra S_i

3. Um classificador final C^* é construído utilizando um esquema de votação ponderada de acordo com o peso de cada classificador; o peso de cada classificador é definido de acordo com sua performance no conjunto de treinamento utilizado para sua construção.

Entrada: *conjunto de treinamento S de tamanho m, Indutor L, número de tentativas T*

1.. $S' = S$ (*exemplos com peso* 1)

2. *Faça* $i = 1$ *até* T {

3. $C_i = L(S')$

4. $e_i = \frac{1}{m} \sum_{x_j \in S' : C_i(x_j) \neq y_j} peso(x)$ (*erro ponderado no conjunto de treinamento*)

5.. *se* $e_i > \frac{1}{2}$, S' *será amostra bootstrap de S com peso* 1 *para cada exemplo e vai para passo* 3

(*este passo é* lim *itado a* 25 *vezes após isto sai do loop*)

6.. $\beta_i = e_i / (1 - e_i)$

7. *Para cada* $x_j \in S'$, *divide peso*(x_j) *por* 2 *se* $C_i(x_j) \neq y_j$, *caso contráio, divide por* $2.(1 - e_i)$

8. *Normaliza os pesos dos exemplos de forma que o peso total de* S' *é m*

9. }

10. $C^*(x) = \arg\max_{y \in Y} \sum_{i : C_i(x) = y} \log \frac{1}{\beta_i}$

Saída: *classificador* C^*

Figura 4 - Algoritmo *Adaboost* baseado em (Bauer and Kohavi,1999)

2.2.3 Random Subspace

O método Random Subspace é uma técnica de combinação de modelos onde os dados de treinamento são amostrados no espaço de características: os classificadores são treinados sobre subespaços aleatoriamente escolhidos do espaço original de treinamento. A definição a seguir é baseada em (Skurichina and Duin, 2002): seja o conjunto de exemplos de treinamento $X = (X_1, X_2,...,X_n)$ onde cada objeto de treinamento X_i ($i=1,...,n$) é um vetor p-dimensional $X_i = (x_{i1},x_{i2},...,x_{ip})$ descrito por p características. Neste método, seleciona-se $r < p$ características

do conjunto de dados *p*-dimensional *X*. É obtido um *r*-dimensional subespaço randômico do espaço *p*-dimensional original de características. Portanto, o conjunto de treinamento modificado $\tilde{X}^b = (\tilde{X}_1^b, \tilde{X}_2^b, ..., \tilde{X}_n^b)$ consiste de exemplos de treinamento r-dimensionais $\tilde{X}_i^b = (x_{i1}^b, x_{i2}^b, ..., x_{ir}^b)$ $(i=1,...,n)$, onde *r* componentes x_{ij}^b $(j=1,...,r)$ são aleatoriamente selecionados de *p* componentes x_{ij} $(j=1,...,p)$ do vetor de treinamento *Xi* (a seleção é a mesma para cada vetor de treinamento). Então, classificadores são construídos a partir dos subconjuntos aleatórios $\tilde{X}^b$ e são combinados através de votação simples majoritária. O método Random Subset é organizado da seguinte forma:

1. *Faça b = 1 até, B*

1.1 Seleciona um sub-espaço randômico r-dimensional $\tilde{X}^b$ do espaço original de características *p-dimensional X*

1.2 Constrói um classificador $C^b(x)$ em $\tilde{X}^b$

2. Combina os classificadores $C^b(x)$, $b=1,2,...,B$, através de votação simples majoritária para uma decisão final.

Figura 5 – Algoritmo Random Subspace (baseado em Skurichina e Duin)

2.3 Integração entre membros de um conjunto de classificadores

A integração de classificadores de um conjunto é feita de forma a tentar classificar um novo exemplo da melhor forma possível. Segundo Puuronen et al. (1999), uma técnica de integração pode resolver o problema de estimativas e seleção dos classificadores mais apropriados para um conjunto. Os métodos de classificação utilizando conjuntos apresentados mostram que a aprendizagem pode ser dividida em duas fases principais: a geração e a integração. Além disso, as técnicas para integração de classificadores podem ser divididas em abordagens estáticas e dinâmicas.

As abordagens estáticas selecionam um modelo para o espaço de dados inteiro ou combina os modelos uniformemente. Estas abordagens são subdivididas entre aquelas que combinam os resultados dos classificadores de um conjunto (fusão de classificadores) e

aquelas que selecionam o melhor classificador do conjunto (seleção de classificador). No primeiro grupo as predições individuais são combinadas numa decisão final. Neste grupo encontram-se as técnicas mais simples e comuns: votação simples (ou majoritária) onde a predição final do conjunto é a predição mais frequente entre os classificadores – caso do *bagging* - e votação ponderada, onde cada classificador tem um peso diferente na votação final, proporcional à habilidade do classificador de classificar novos exemplos – caso do *boosting*. No segundo grupo, dentre as técnicas que selecionam o melhor classificador, a mais simples e mais popular é a Validação Cruzada[5] Majoritária (CVM - *Cross-Validation Majority*), onde a taxa de acerto de validação cruzada de cada classificador é estimada com os dados de treinamento. No final, o classificador com o melhor desempenho é selecionado (Yu-Quan, et.al,2011).

Menos populares que as estáticas, as abordagens dinâmicas recebem esta denominação pelo fato de levar em conta as características do exemplo a ser classificado. Moreira (2009) divide esta abordagem nas seguintes tarefas: (i) dado um exemplo de entrada x, procura-se similares em um conjunto com dados para validação; (ii) seleciona o(s) classificador(es) do conjunto que tenha(m) melhor desempenho nos dados selecionados para validação; (iii) obtém as predições do(s) classificador(es) selecionado(s) para o exemplo de entrada; (iv) obtém uma predição global do conjunto. Neste tipo de abordagem geralmente são utilizados três conjuntos de dados: de treinamento, de validação e de teste. A combinação das predições individuais pode ser feita através de métodos de votação dinâmicas onde o peso de cada classificador é proporcional a seu desempenho em dados similares. Para combinar as predições em uma predição global algumas metodologias dinâmicas serão apresentadas a seguir.

Puuronen et al. (1999) apresenta uma técnica dinâmica que utiliza estimativas de erros de classificação: durante a fase de treinamento, informações sobre a qualidade dos classificadores é coletada em uma matriz de desempenho. Durante a fase de classificação, esta matriz é utilizada para prever a qualidade de predição produzida pelos classificadores para um novo exemplo. Na técnica proposta em (Tsymbal, 2005) o peso de cada classificador será dado de acordo com seu desempenho em exemplos vizinhos do exemplo a ser classificado, e não de acordo com sua precisão global. Yu-Quan et.al (2011) apresenta o método DWEC-CV (*Dynamic Weighting Ensemble Classifiers based on Cross-Validation*) que emprega a técnica

[5] Validação cruzada (*Cross-validation*) é um método estatístico para avaliação e comparação de algoritmos de aprendizagem através da divisão dos dados em dois conjuntos: um usado para aprendizagem (ou treinamento) do modelo e o outro usado para validar o modelo.

de validação cruzada para, dinamicamente, atribuir um peso a cada classificador; o peso de cada classificador é utilizado na votação para a predição do conjunto.

De forma geral, as técnicas dinâmicas apresentaram resultados favoráveis quando comparados com técnicas estáticas de integração (Puuronen et al.,1999).

2.4 Drift detection

Um grande problema para algoritmos de aprendizagem é que, em muitos domínios do mundo real, os conceitos não são estáveis e mudam com o tempo. Nestes domínios, o conceito de interesse pode depender de um contexto que não é explicitado na forma de características preditivas, denominado "contexto escondido". Um exemplo típico é a previsão do tempo, que pode variar radicalmente com a estação. Outro exemplo é o padrão de preferência de consumidores que mudam de acordo com o panorama econômico, existência de outros fornecedores, etc. As mudanças escondidas neste contexto podem induzir mudanças mais ou menos radicais no conceito, de forma que o modelo construído sobre os exemplos de treinamento torna-se inconsistente com os novos exemplos que devem ser classificados. Este problema conhecido como *concept drift* complica a tarefa de aprendizagem de um modelo e requer abordagens especiais, diferentes de técnicas normalmente usadas, que trata novos exemplos como igualmente importantes para o conceito final. O trabalho de Tsymbal (2005) apresenta um exemplo típico de *concept drift* no contexto médico: a resistência aos antibióticos. Neste domínio, a sensibilidade de um micro-organismo infeccioso pode mudar ao longo do tempo, de forma que novas cepas do micro-organimo podem desenvolver resistência aos antibióticos aos quais eram sensíveis anteriormente.

A Figura 6 ilustra uma mudança de conceito, definido em (Stanley, 2001) da seguinte forma: sejam dois conceitos *A* e *B*. Uma sequência de exemplos $i_1....i_n$ é apresentada para o algoritmo de mudança de conceito. Antes de um exemplo i_x, o conceito objetivo *A* é estável e não muda. Após um número de exemplos Δx após i_x, o conceito estável é *B*. Entre os exemplos i_x e $i_x+\Delta x$, o conceito está mudando entre os objetivos *A* e *B* de acordo com alguma distribuição.

Quando $\Delta x = 1$, a mudança entre *A* e *B* é súbita (ou abrupta). Quando $\Delta x > 1$, o conceito está mudando sobre um certo número de exemplos, de forma gradual. Um exemplo para ilustrar o primeiro caso seria uma situação onde um recém-formado na universidade, muda

completamente suas preocupações financeiras. Por outro lado, o desgaste provocado pelo uso de um equipamento de fábrica pode causar uma mudança gradual na qualidade das peças produzidas.

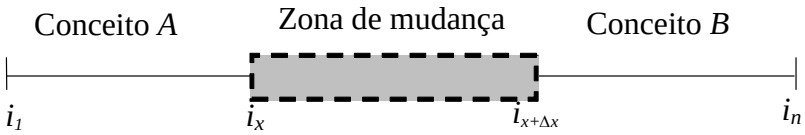


Figura 6 – Mudança do conceito *A* para *B* (Enembreck et al., 2009)

Segundo Tsymbal (2004) um domínio com *concept drift* requer abordagens especiais, diferentes de técnicas comumente usadas, onde os novos exemplos desempenham papel importante no conceito final. Para ele, um sistema ideal para lidar com mudança de conceito deve:

(1) Detectar mudanças de conceito e se adaptar rapidamente a elas. O sistema precisa de um *feedback* para suas tentativas de classificação e sempre que a precisão de predição (ou desempenho do sistema) diminuir significa que o conceito está mudando;

(2) Ser robusto em relação a ruído e conseguir distingui-lo de uma mudança de conceito. O algoritmo pode ser altamente sensível a ruído, interpretando-os como mudança de conceito ou, por outro lado, pode ser muito conservador e demorar a se adaptar a esta mudança;

(3) Conseguir reconhecer e tratar contextos recorrentes: contextos recorrentes podem ocorrer devido a fenômenos cíclicos, como as estações do ano. Descrições de conceitos podem ser salvas neste domínio para que estas descrições possam ser reexaminadas e reusadas mais tarde, o que tornaria a adaptação do sistema mais rápida.

De acordo com a abordagem utilizada, Tsymbal (2004) classifica os sistemas disponíveis que lidam com *concept drift* em: (1) **Seleção de exemplo** (*windowing*) que é a técnica mais comum para lidar com mudança de conceito, onde o aprendiz utiliza somente um conjunto com os últimos exemplos – este conjunto recebe o nome de janela – para criar o modelo de dados corrente. À medida que novos exemplos são adicionados à janela, os mais velhos são excluídos. No modelo mais simples, a janela é de tamanho fixo e o exemplo mais

velho é eliminado na chegada de um novo exemplo (*time-base forgetting*) e nos modelos mais elaborados, utiliza-se uma heurística para definir o tamanho da janela. Exemplos de algoritmos baseados em janelas são os algoritmos da família FLORA (Widmer e Kubat, 1996). O algoritmo FLORA2, por exemplo, utiliza uma heurística para adaptar o tamanho da janela dinamicamente durante o processo de aprendizagem. Já o algoritmo FLORA3 é uma extensão de FLORA2, que inclui o mecanismo para armazenamento de contextos anteriores. Um outro método, proposto por Gama (2004), utiliza o desempenho do algoritmo para verificar a presença de *concept drif*. Neste método são estabelecidos dois níveis de erro, um nível de alerta e um nível de *drift.* Quando inicia uma mudança, o nível de erro aumenta, atingindo o nível de alerta. No momento (num determinado exemplo) em que atinge o nível de *drift*, o algoritmo aprende um novo modelo a partir deste exemplo; (2) **Exemplo ponderado** é uma técnica que usa a habilidade de alguns algoritmos como máquinas de vetor de suporte (SVM – *Support Vector Machines)* (Cristianini e Shawe-Taylor, 2000) para processar exemplos ponderados pela idade e pelo desempenho em relação ao conceito corrente. Parte da ideia de que a importância de um exemplo deve decrescer com o tempo; (3) **Aprendizagem por conjunto** é uma técnica baseada nos métodos de classificação que utilizam um conjunto de classificadores (Seção 2.2), onde a classificação final é dada pela combinação de votos destes classificadores. Todas as abordagens que utilizam conjunto de classificadores utilizam algum critério para excluir, reativar ou criar novos membros do conjunto, que é normalmente baseado em consistência do modelo com os dados correntes. O modelo se torna inconsistente quando ocorre uma mudança de conceito e, consequentemente um decréscimo no desempenho dos classificadores. Os classificadores com pior desempenho são, então, substituídos até que o modelo se torne consistente novamente. Outra característica desta técnica é a forma como a integração dos classificadores é feita para obtenção da decisão do conjunto: normalmente através de votação ponderada com o peso de cada classificador proporcional a seu desempenho individual.

Alguns algoritmos de aprendizagem utilizados como base de modelos nos sistemas que lidam com *concept drift* são: **aprendizagem baseado em regras**, utilizada nos algoritmos da família Flora (Widmer e Kubat, 1996); **árvores de decisão**, utilizadas no algoritmo DWM-Dynamic Weighted Majority (Kolter e Maloof, 2003); **aprendizagem bayesiana**, também utilizado no DWM; SVM, utilizado nos métodos apresentados em (Klinkenberg, 2004); **redes de funções de base radial**, aplicadas no sistema FRANN (Kubat e Widmer, 1995); **aprendizagem baseada em exemplo**, utilizada no framework e metodologia *Instance Based Learning* proposta em (Aha, et al.,1991).

A maioria dos algoritmos de aprendizagem é incapaz de se adaptar a um tipo de mudança conhecida como mudança de conceito local, que é muito comum com dados do mundo real. Tsymbal et al. (2005) argumenta que uma mudança de conceito local ocorre entre dois tempos consecutivos t_1 e t_2 se existe um subconjunto *X'* dentro do conjunto de exemplos *X* tal que *X'* tem diferentes mudanças de conceito ou distribuição de dados em comparação com o resto dos dados. Ou seja, uma mudança de conceito local indica mudanças no conceito ou na distribuição de dados que ocorrem em algumas partes do conjunto de exemplos. Quando este tipo de mudança ocorre, muitos modelos globais são descartados porque os desempenhos destes com os dados atuais diminui, mesmo que estes modelos tenham bom desempenho nos dados que apresentam estabilidade. Segundo Tsymbal (2004) esta limitação é um problema típico de aprendizes ansiosos. Os algoritmos de aprendizagem preguiçosos, por sua natureza, se adaptam bem a mudança de conceito local. Delany et al. (2005) apresenta as vantagens dos algoritmos preguiçosos quando estes são aplicados num filtro de *spam*, domínio onde alguns tipos de *spam* mudam com o tempo, enquanto outros não se alteram. Estas vantagens são: (1) bom desempenho com conceitos muitos diferentes (por exemplo, um *spam* oferecendo remédios baratos é muito diferente de um *spam* oferecendo empréstimos) porque não tenta aprender um conceito unificado; (2) casos-base são fáceis de atualizar, pois quando surgem novos tipos de *spam*, não é necessário reconstruir o modelo, somente a atualização do caso-base com novos exemplos; (3) permite compartilhar conhecimento para determinados problemas, permitindo a manutenção de múltiplos casos-base potencialmente distribuídos.

2.4.1 Sistemas que utilizam a técnica de conjunto de classificadores

Entre as abordagens mais populares e efetivas para lidar com *concept drift* está a aprendizagem por conjunto (*Ensemble Learning*). Alguns sistemas que utilizam esta abordagem serão apresentados a seguir:

A. *Concept Drift Committee* (CDC)

O método chamado CDC (*Concept Drift Committee*) foi apresentado em (Stanley, 2003). Neste método, um comitê de tamanho fixo de árvores de decisão incrementais é treinado, sendo que todas as árvores são atualizadas quando um novo exemplo de treinamento torna-se disponível. Num determinado instante do tempo, o desempenho dos membros é estimado de acordo com uma janela de tamanho fixo. Membros de baixo desempenho são excluídos e substituídos por uma nova árvore treinada com todos os exemplos subsequentes. O algoritmo funciona da seguinte forma:

- Um comitê *C* é composto de um máximo de *n* hipóteses ($h_1 ... h_n$), sendo cada hipótese uma árvore de decisão;

- O comitê é iniciado vazio. Sempre que um novo exemplo de treinamento é processado e o comitê tem menos que *n* hipóteses, um novo membro é adicionado no comitê. Este novo membro é iniciado somente com o exemplo de treinamento corrente. Os demais membros já existentes são também treinados com este exemplo corrente.

- Após atingir um número máximo *n* de membros, novos membros somente podem ser adicionados se algum membro antigo for retirado.

Para que os membros do comitê sejam avaliados, cada membro deve votar em exemplos de teste, derivados da distribuição corrente. O peso do voto de cada membro é igual ao seu desempenho do membro nos últimos *m* exemplos de treinamento. Portanto, membros com desempenho satisfatório possuem votos com maior peso. Para que o comitê seja mantido atualizado com o conceito corrente, os membros devem ser substituídos por novos membros no momento em que sua performance cai abaixo de um *threshold t*. Quando criado, um novo membro não tem conhecimento dos exemplos anteriores à sua criação e por isto o método CDC permite que o este membro atinja um grau de maturidade até que ele possa votar e ser substituído, quando necessário. Este grau de maturidade é atingido quando o membro já observou *n* exemplos (o grau de maturidade é igual ao tamanho do comitê para que, na pior situação, quando todos os membros do comitê tiverem um mau desempenho, sempre haverá pelo menos um membro maduro para ser retirado).

Na prática, o comitê se torna maduro como um todo e se mantém estável quando o conceito alvo não está mudando. Quando o conceito muda, há muitos membros sendo retirados e substituídos. Uma mudança abrupta de conceito geralmente provoca a substituição de quase todos os membros do comitê. No caso de uma mudança gradual, os membros são substituídos gradualmente. Estes comportamentos fazem com que este algoritmo seja tolerante a ruído, e eficiente para detecção de *concept drift*.

Nas avaliações experimentais, Stanley constatou que em uma situação de mudança abrupta, CDC e FLORA4 (o melhor dos algoritmos da família FLORA) possuem desempenho similar. Na situação de mudança moderada, CDC é um pouco mais exato que FLORA4. A maior diferença ocorre na mudança gradual de conceito, onde CDC tem vantagem em relação ao FLORA4.

B. *Streaming Ensemble Algorithm* (SEA)

Da mesma forma que o CDC, o SEA (Street e Kim, 2001) é um algoritmo baseado num comitê de classificadores que utilizam árvores de decisão. O SEA constrói classificadores em porções sequenciais dos dados de treinamento. Os classificadores são combinados em um conjunto de tamanho fixo usando uma heurística na estratégia de substituição.

O *framework* para construção do comitê pode ser resumido da seguinte forma: classificadores individuais são construídos a partir de subconjuntos de dados, processados em blocos sequenciais. Os classificadores são então combinados em um comitê de tamanho fixo. Quando o comitê está completo, novos classificadores somente serão adicionados se estes melhoram o desempenho do comitê. Neste caso, para que o tamanho do comitê seja mantido constante, um classificador deve ser removido, mantendo o tamanho do conjunto constante. Estimativas de desempenho são feitas testando o novo modelo (e o modelo existente) no próximo bloco de dados. A chave do desempenho deste algoritmo é o método usado para determinar se um novo modelo deve ser adicionado ao conjunto e qual modelo deve ser removido. O método utilizado favorece os classificadores que classificam corretamente quando o conjunto está indeciso. Os resultados dos experimentos do algoritmo SEA sobre conjuntos de dados reais podem ser encontrados em (Street e Kim, 2001). De maneira geral, o algoritmo se mostra rápido para dados em grande escala, classifica tão bem quanto uma árvore de decisão construída sobre todos os dados, no entanto requer memória constante e se adapta rapidamente a mudança de conceito.

C. *Dynamic Weighted Majority* (DWM)

Ao contrário dos métodos anteriores, onde as árvores de decisão foram utilizadas como algoritmo base de aprendizagem pelos membros do comitê, o DWM (*Dynamic Weighted Majority*) é um método geral, onde um algoritmo *master* usa as predições de um conjunto de algoritmos de predição para fazer sua própria predição. O objetivo é que o algoritmo *master*

produza menos erros que o melhor algoritmo do conjunto. DWM é baseado no algoritmo *Weighted Majority (*Littlestone e Warmuth, 1994) (maioria ponderada), onde mecanismos para criação e exclusão de classificadores (chamados *experts*) foram adicionados e foi apresentado em (Kolter e Maloof, 2003). A Figura 7 representa o algoritmo DWM que funciona da seguinte forma:

- Mantém um conjunto de classificadores com um peso associado.

- Quando um novo exemplo é apresentado, o elemento de performance (algoritmo *master*) passa o exemplo para todos os classificadores, que retornam a predição para o exemplo. Se um classificador prediz incorretamente (ou seja, classifica diferente da classe indicada no exemplo), seu peso é diminuído.

- Usando as predições e os pesos dos membros, DWM determina uma predição global. Se esta predição é incorreta, o algoritmo cria um novo classificador com peso *1* e normaliza o peso dos classificadores existentes de forma que o maior peso seja igual a *1* (isto para evitar que um novo classificador domine a decisão).

- O algoritmo também remove todos os classificadores com peso menor que um *threshold* definido.

- No final, passa o exemplo novamente para todos os classificadores, para que sejam treinados.

A normalização dos pesos e o treinamento incremental dos classificadores fazem com que eles se recuperem da mudança de conceito (*concept drift*). O algoritmo utiliza também um parâmetro que indica a frequência com que os pesos são normalizados e os classificadores são adicionados e removidos.

Os estudos empíricos de Kolter e Malof (2003) sugerem que, quando comparado com outros métodos, DWM manteve um número comparável de classificadores porém atingiu maior precisão nas predições e convergiu para esta precisão mais rapidamente.

Uma aplicação que utiliza o DWM na coordenação de classificadores é apresentada em (Enembreck et al., 2009) cuja proposta é que, em um ambiente de negociação, agentes inteligentes de negócio tentam identificar mudanças na estratégia de negociação de seus opositores através do *drift detection*. Para isto, um agente comprador utiliza um comitê de classificadores (que utilizam o algoritmo IB3 como algoritmo base de aprendizagem) para tentar descobrir a descrição de uma oferta interessante.

Dynamic Weighted Majority $(\{\vec{x}, y\}_n^1, c, \beta, \theta, p)$

$\{\vec{x}, y\}_n^1$	: dados de treinamento, vetor de características e classificação
$c \in \mathrm{N}^*$	: quantidade de classes, $c \geq 2$
β	: fator de decréscimo do peso, $0 \leq \beta < 1$
θ	: *threshold* para remoção de classificadores (*experts*)
p	: período entre remoção, criação e atualização do peso dos *experts*
$\{e, w\}_m^1$	: conjunto de *experts* e seus pesos
$\Lambda, \lambda \in \{1,...,c\}$	: predição global e individual
$\vec{\sigma} \in \Re^c$	: soma das predições ponderadas para cada classe

1. $m = 1$
2. e_m = *Cria-Novo-Expert()*
3. w_m=1
4. Para $i = 1$ até n Faça { (*loop* para os exemplos)
5. $\vec{\sigma} = 0$
6. Para $j = 1$ até m Faça { (*loop* para os *experts*)
7. λ = *Classifica* (e_j, $\vec{x}_i$)
8. Se ($\lambda \neq y_i$ e $i \bmod p = 0$), $w_j = \beta w_j$
9. $\sigma_\lambda = \sigma_\lambda + w_j$
10. }
11. $\Lambda = arg\ max_j\ \sigma_j$
12. Se ($i \bmod p = 0$) {
13. w = Normaliza-Pesos(w)
14. $\{e, w\}$ = Remove-Experts ($\{e,w\}$,θ)
15. Se ($\Lambda \neq y_i$) {
16. m = m + 1
17. e_m = *Cria-Novo-Expert()*
18. w_m=1
19. }
20. }
21. Para $j = 1$ até m, e_j = *Treinar*(e_j , $\vec{x}_i$, y_i)
22. Saída: Λ
23. }
24. Fim

Figura 7 - Algoritmo DWM baseado em (Kolter e Maloof, 2003)

D. *Additive expert ensembles* (AddExp)

Apresentado pelos mesmos pesquisadores que desenvolveram o DWM, o algoritmo AddExp possui muitas similaridades com DWM: mantém um conjunto de classificadores

(denominados *experts*), criados em momentos diferentes e com o mesmo algoritmo de treinamento e predição. O AddExp generaliza também para tarefas de regressão e possui mecanismos que limitam o número de classificadores.

O elemento de performance do algoritmo utiliza a votação ponderada dos classificadores. Para cada classificação, o algoritmo soma os pesos de todos os membros que predizem aquela classificação e prediz a classificação com o maior peso (a única diferença entre AddExp para classes discretas e classes contínuas é que as classes preditas no primeiro caso pertencem a um conjunto discreto de classes e no segundo caso, estão no intervalo *[0,1]*). Primeiro, os *experts* predizem a classificação do exemplo de treinamento. Os pesos de todos os classificadores que classificam incorretamente são diminuídos. Se a predição global for incorreta, um novo classificadore é adicionado. No final, todos os classificadores são treinados com o exemplo. O algoritmo AddExp pode ser consultado em (Kolter e Maloof, 2005).

AddExp também pode utilizar duas técnicas de poda (remoção de classificadores). Na técnica *Oldest First*, quando novo membro é adicionado ao conjunto, se o número de membros é maior que um valor *K*, o membro mais velho é removido para adição do novo. Na técnica *Weakest First*, ao invés do membro mais velho, é removido o membro com menor peso.

2.5 Análise de redes sociais

Nas seções anteriores foram apresentados sistemas de aprendizagem que utilizam a abordagem onde vários classificadores são agrupados, formando um conjunto que será o responsável pela predição do sistema. Nesta seção será feita uma introdução ao paradigma de Análise de Redes Sociais apresentando um breve histórico, os modelos de rede utilizados e conceitos e métricas deste paradigma. Esses conceitos são fundamentais para a compreensão da abordagem social de combinação de classificadores proposta nas seções seguintes deste documento.

2.5.1 Histórico

Wasserman e Faust (1994) definem uma rede social como um conjunto finito ou conjuntos de atores e a relação ou relações definidas entre eles sendo que a presença de informação das relações é uma característica fundamental. Para Knoke (2008) uma rede social é uma estrutura composta por atores conectados entre eles por um ou mais tipos de relação. Os nós podem representar pessoas, entidades, organizações, sistemas, etc., que podem ser analisados individualmente ou coletivamente, observando a relação entre eles (a relação pode representar amizade, conhecimento profissional ou proximidade geográfica).

A Análise de Redes Sociais (ARS) é um paradigma que utiliza a teoria de redes para medir e representar a estrutura de relações entre os atores, explicar porque tais relações existem e quais as suas consequências. Segundo Knoke e Yang (2008) o que diferencia a ARS é que, enquanto a maior parte das ciências sociais assume que os atores agem e tomam decisões sem levar em consideração o comportamento de outros atores, a ARS assume que os atores fazem parte de um sistema social que os conecta a outros atores, e cujas relações exercem influências importantes no comportamento de cada ator. O foco da ARS são as relações e os padrões das relações entre os atores (os atributos dos atores ficam em um segundo plano) e isto requer um conjunto de métodos e conceitos analíticos diferentes dos métodos estatísticos tradicionais de análise de dados. Um analista de rede deve procurar modelar as relações para descrever a estrutura de um grupo e, a partir daí, estudar o impacto da estrutura sobre o funcionamento do grupo e/ou a influência destra estrutura sobre os indivíduos dentro do grupo (Wasserman e Faust, 1994).

Para Knoke e Yang (2008) a importância de ARS está nas seguintes suposições sobre padrões de relações e seus efeitos: (i) relações estruturais geralmente são mais importantes para entender comportamentos observados do que atributos como idade, gênero, valores e ideologia; (ii) redes sociais afetam percepções, crenças e ações através de uma série de mecanismos estruturais que são construídos através de relações entre entidades; (iii) por último, relações estruturais devem ser vistas como processos dinâmicos, ou seja, não são estruturas estáticas mas estão continuamente mudando através de interações entre os atores.

No princípio, a ARS era utilizada na sociologia e antropologia, porém, com o desenvolvimento das ferramentas matemáticas e computação, começou a avançar em diversas áreas como computação, matemática, física, saúde, economia, entre outros, aplicada em diversos domínios como Internet, disseminação de vírus, movimentos sociais, redes de

terrorismo, redes de distribuição (como vasos sanguíneos ou rotas de correio, por exemplo), teias alimentadoras, etc. (Newman, 2003).

2.5.2 Teoria dos grafos

A teoria dos grafos fornece uma representação de uma rede social e um conjunto de conceitos que podem ser utilizados para o estudo formal das propriedades das redes sociais: desta forma em termos matemáticos, uma rede pode ser representada por um grafo.

A teoria dos grafos tem origem no século XVIII, no trabalho de Leonhard Euler. Um grafo é definido em (Szwarcfiter, 1986) como um conjunto finito não-vazio *V* e um conjunto *E* de pares não-ordenados de elementos distintos de *V*. Os elementos de *V* são os vértices, os de *E* são as arestas de *G* e o grafo é representado por *G(V,E)*. Cada aresta *e* pertencente a *E* será denotada pelo par de vértices *e = (v,w)* que a forma. Neste caso, os vértices *v* e *w* são os extremos da aresta e, sendo denominados adjacentes. A aresta e é dita incidente a *v* e *w*. Duas arestas que possuem um extremo comum são chamadas adjacentes. No caso de uma rede social, os vértices do grafo representam os atores e as arestas representam as relações entre eles.

A definição de alguns termos utilizados na teoria de redes será importante para a continuidade do trabalho como: (1) *grau*, que indica o número de arestas conectadas a um nó; (2) *componente* ao qual pertence um nó, que é definido como um conjunto de nós que podem ser alcançados a partir do nó, através dos caminhos ao longo das arestas; (3) *caminho geodésico*, nome dado ao menor caminho de um nó a outro através da rede; (4) *diâmetro* que é definido como o comprimento (em número de nós) do maior caminho geodésico entre dois nós dentro da rede.

As relações entre os atores possuem algumas propriedades como: (1) *valoração*, que indica se a relação é dicotômica (ou binária), ou seja, existe ou não existe ou se a relação é valorada, quando possui um valor discreto ou contínuo; (2) *direcionamento*, que indica se a relação é direcional (dígrafo), onde um ator é o transmissor e o outro é receptor (por exemplo, uma rede representando *emails* entre indivíduos), ou não-direcional, onde a relação é recíproca. No caso de um dígrafo, os termos grau de entrada e grau de saída são utilizados para indicar os números de arestas que chegam e que saem de um nó, respectivamente. Os dígrafos podem ser cíclicos (quando o grafo contém *loops* de nós) ou acíciclos.

2.5.3 Modelos de redes

Os modelos de redes propostos na literatura surgiram da necessidade de entender os mecanismos de surgimento e evolução de redes complexas como a WWW (onde os nós são os documentos e as arestas são os *hiperlinks*), a Internet (pode ser estudado tanto no nível roteador, onde cada nó é um roteador, ou no nível interdomínio, onde cada domínio representa um nó), redes de colaboração científica, redes ecológicas, rede de chamadas telefônicas, rede de citações e redes neurais e de potência.

Um modelo de rede bastante utilizado em estudos é o modelo de grafos randômicos, onde os vértices são distribuídos randomicamente. Apresentado em (Erdös e Rénvi, 1958), neste modelo os nós se conectariam aleatoriamente e as redes seriam igualitárias, pois todos os nós teriam a mesma probabilidade de receber uma conexão e possuiriam aproximadamente o mesmo número de conexões. Porém, este modelo possui limitações. Por exemplo, a probabilidade que duas pessoas que tenham um amigo em comum se conhecerem é bem maior que duas pessoas escolhidas aleatoriamente se conhecessem.

No início dos anos 60, Stanley Milgram conduziu um experimento para observar a probabilidade de que duas pessoas escolhidas aleatoriamente se conhecessem: cartas foram enviadas aleatoriamente a vários indivíduos, pedindo que eles a enviassem a um destinatário específico não conhecido. Os indivíduos deveriam então enviar as cartas para outras pessoas que acreditassem que pudessem conhecer este destinatário. O resultado foi que, todas as cartas que chegaram ao destinatário haviam passado por um pequeno número de pessoas (em média 6 pessoas). Isso indicaria que essas pessoas estariam a poucos graus de separação umas das outras. Daí vem o termo "mundo pequeno". Esse modelo pode ser especialmente aplicado às redes sociais: cada indivíduo tem amigos e conhecidos em todo o mundo, que por sua vez, conhecem outras pessoas. Assim, todos estaríamos "conectados", o que evidenciaria a existência de poucos graus de separação entre as pessoas no planeta. Desta concepção da sociedade humana deriva o conceito de redes de Mundos Pequenos (*Small World*) formalizado por Watts e Strogatz em 1998. O modelo Watts-Strogatz ainda se enquadra no modelo de grafos aleatórios, porém tenta combinar características dos grafos aleatórios e dos grafos regulares (que possuem vértices com o mesmo número de vértices adjacentes). Enquanto os grafos aleatórios apresentam um coeficiente de agrupamento (média de pares de vértices ligados a um dado vértice e que são ligados entre si) baixo quando comparado com os grafos regulares, a distância média entre vértices (*path length*), o caminho entre dois vértices, aumenta nos grafos regulares com a quantidade de nós enquanto é muito menor nos grafos

aleatórios. O modelo de Mundos Pequenos apresenta o coeficiente de agrupamento similar ao dos grafos regulares e o comprimento do caminho similar ao dos grafos aleatórios e pode representar muitas redes reais. Porém tal modelo não engloba, por exemplo, a WWW, que é uma rede aberta, onde novos vértices (páginas *web*) estão sempre sendo criados, mas que não são conectados aleatoriamente a vértices existentes.

As redes reais, cuja distribuição de grau, diferentemente dos grafos randômicos, segue uma lei de potência (ou seja, alguns nós tendem a receber muito mais conexões que outros) recebe o nome de redes livres de escala, uma vez que leis de potência são livres de uma escala característica. Para descobrir o mecanismo de surgimento de uma rede livre de escala, Albert e Barabási (2002) propuseram uma modelagem que tenta capturar a dinâmica de uma rede, ou seja, o processo de construção e evolução da rede. O modelo foi inspirado por dois mecanismos genéricos presentes em muitas redes reais: (i) as redes descrevem sistemas abertos, que crescem através da adição contínua de novos nós e (ii) anexação preferencial (a probabilidade de um nó receber uma nova conexão depende do grau – ou conectividade - deste nó). O algoritmo de Albert-Barabási pode ser descrito da seguinte forma:

(1) Crescimento: começando com um pequeno número de nós (m_0), a cada passo, um novo nó é adicionado com m ($m \leq m_0$) arestas que ligam o novo nó com m diferentes nós presentes ao sistema.

(2) Anexação preferencial: a escolha dos nós aos quais o novo nó será anexado é feita utilizando uma probabilidade, que depende do grau de cada nó passível de escolha. Ou seja, a probabilidade de que um novo nó se conecte a um nó i depende do grau k_i do nó i, de forma que a probabilidade Π seja dada pela Fórmula 3.

$$\Pi(i) = \frac{k_i}{\sum_j k_j}$$

Fórmula 3 – Probabilidade de que um novo nó se conecte a um nó *i*

A Figura 8 representa uma ilustração dos possíveis processos elementares para $m_0 = 3$ e $m=2$. Inicialmente composta por m_0 nós, quando novos nós são adicionados m relações que partem destes novos nós são criados. Os nós destinos destas relações são selecionados através

da anexação preferencial, ou seja, através de uma probabilidade proporcional à centralidade de cada nó destino possível.

Uma característica de rede livre de escalas são os *hubs*: alguns vértices que concentram uma grande quantidade de ligações. A topologia de páginas WEB (onde os nós são as páginas e as ligações são os *hiperlinks*) e de artigos científicos (onde os nós são as publicações e as relações são as citações) são exemplos de redes que exibem esta característica.

O modelo Albert-Barabási é um modelo mínimo que representa o mais simples mecanismo de auto-organização de uma rede em crescimento em uma estrutura livre de escala e, por isto, possui limitações. Alguns estudos, apresentados a seguir, tentam descobrir os mecanismos genéricos de crescimento de redes reais.

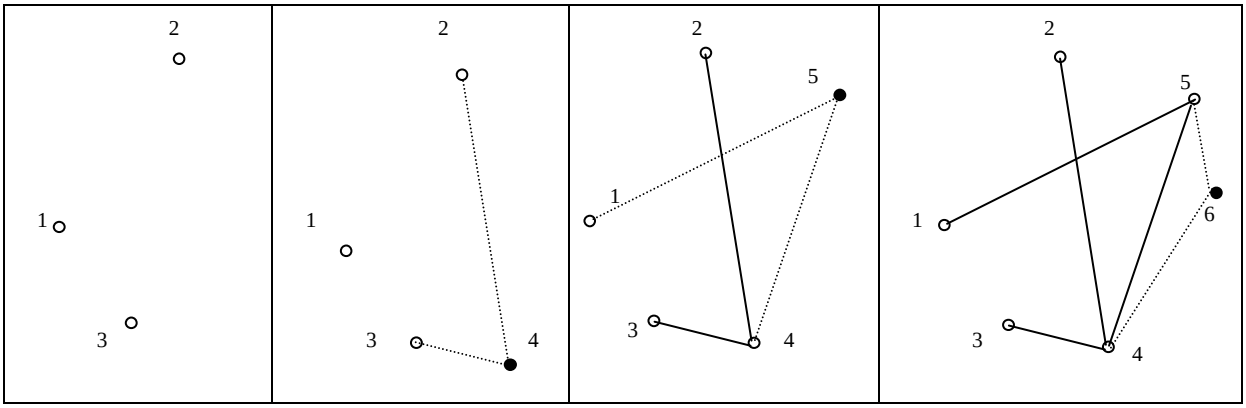


Figura 8 – Criação de rede livre de escalas baseado em (Albert e Barabási, 2000)

Em (Dorogovstsev at al, 2000) é apresentado um modelo de crescimento de rede que generaliza o modelo Alberto-Barabási levando em conta uma "atratividade inicial". Neste modelo, a probabilidade de que um novo nó seja anexado a um nó existente é a soma de uma atratividade inicial com o grau do nó que vai receber a conexão. Esta atratividade representa a chance que um nó em redes reais possui, mesmo que isolado, de receber uma conexão. Albert e Barabási (2000), que no seu modelo original consideram somente a adição de nós, incorporam eventos locais que regem a evolução da rede – adição e remoção de nós ou adição e remoção de vértices. Este estudo mostra que, dependendo da frequência destes eventos, uma rede pode desenvolver um modelo diferente de topologia: a distribuição de conectividade pode seguir uma lei de potência generalizada ou uma exponencial.

Uma característica estudada em (Dorogovstsev e Mendes, 2000) é o tempo de vida dos nós através da evolução de redes de referência, uma vez que novos artigos raramente citam

artigos mais antigos. O trabalho sugere que em alguns sistemas a probabilidade que um nó existente receba uma nova conexão decai com a idade do nó. Bianconi e Barabási (2001) propõem um modelo que permite investigar o aspecto competitivo de alguns sistemas como sistemas sociais, redes de citações e páginas *web*, onde conectividade e crescimento dependem também de uma habilidade (*fitness*) que cada nó possui. Isto explicaria, por exemplo, porque algumas páginas web adquirem *links* rapidamente, superando páginas mais antigas. Dorogovtsev et al. (2000) propõe um modelo onde a anexação preferencial é combinada com uma herança parcial da conectividade de um nó existente, pelo novo nó. É assumido que cada novo nó nasce a partir de outro nó existente, escolhido randomicamente e herda uma parte da conectividade do nó pai, ou seja, parte das arestas que ligam o nó pai são duplicadas apontando para o novo nó.

Além dos modelos anteriormente apresentados que tentam complementar o modelo de Albert e Barabási, alguns modelos foram propostos como alternativa ao mecanismo de anexação preferencial: (i) *mecanismo de cópia*: motivados pelo desejo de explicar a distribuição de grau de lei de potência (Kleinberg, et al., 1999 apud Albert e Barabási) e (Kumar, et al. 2000 apud Albert e Barabási) assumem que novas páginas WEB sobre um certo assunto copiam *links* de páginas existentes sobre o mesmo assunto. Neste modelo, a cada passo um novo nó é adicionado, e se conecta aos demais nós com um número constante de vértices. Ao mesmo tempo, um nó protótipo é escolhido randomicamente dentre os nós existentes. Os vértices do novo nó são distribuídos da seguinte forma: com probabilidade *p* o destino do *i-ésimo* vértice é selecionado randomicamente e com probabilidade *1-p* é selecionado para ser o destino do i-ésimo vértice do nó protótipo; (ii) *redirecionamento de vértice* (Krapivsky e Redner, 2001 apud Albert e Barabási): neste modelo a cada passo um novo nó é adicionado e um nó é selecionado como um possível alvo de anexação. Com probabilidade (*1-r*) um vértice direcionado do novo nó para *i* é criado; no entanto, com probabilidade *r* o vértice é redirecionado para o nó antepassado *j* do nó *i* (isto é, o nó ao qual *i* é anexado na primeira adição da rede); (iii) *anexação a vértices* (Dorogovtsev, 2001 apud Albert e Barabási): neste modelo, a cada passo um novo nó se conecta aos nós de um vértice randomicamente selecionado. Ou seja, a probabilidade de que um nó vai receber um novo vértice é diretamente proporcional ao seu grau.; em outra palavras, este modelo tem exatamente a mesma anexação preferencial que o modelo Albert-Barabási.

2.5.4 Conceitos fundamentais da análise de redes sociais

Os métodos de análise de rede social podem ser aplicados a vários contextos: pesquisas de propagação de doenças, difusão de tecnologias, análise organizacional, desenvolvimento distribuído de softwares, combate ao terrorismo, etc. A seguir são apresentados alguns conceitos da análise de redes sociais relacionados a métricas locais e topológicas de relevância dos indivíduos que formam a rede.

A. Centralidade

A utilização da teoria de grafos permite definir a importância de um ator dentro da rede através de conceitos como centralidade e prestígio. Para definição destes conceitos será utilizada a representação de uma rede utilizando a teoria de grafos. Tal representação é constituída de: (1) um conjunto de atores (*N*) contendo ***g*** atores e denotado por: $N = \{n_1,n_2,....,n_g\}$; (2) um conjunto (*L*) contendo *nL* pares de atores $l = <n_i, n_j>$ (cada par representando uma relação entre os atores n_i e n_j) denotado por: $L = \{l_1,l_2,...l_{nL}\}$. Para uma rede representada pelo grafo (*N*,*L*), onde *N* contém *g* atores, *L* contém no máximo *g(g-1)* elementos se as relações de *L* são direcionais. Caso contrário, *L = g(g-1)/2*.

A centralidade é medida de acordo com as relações às quais o ator está envolvido, independente se é um transmissor ou um receptor. Calcular a centralidade de um ator consiste em identificar a posição em que ele se encontra em relação às trocas e à comunicação na rede. Quanto mais central é um indivíduo, mais bem posicionado ele está em relação às trocas e à comunicação, o que aumenta o seu poder na rede. Uma outra medida de centralidade pode ser medida por grupo, através da combinação das métricas dos atores e pode ser utilizada para comparar diferentes redes.

As medidas de centralidade mais utilizadas são:

(1) **Centralidade de grau** – a medida de centralidade de grau para um ator individual é o grau do nó, ou seja, o grau de um ator *i* é a soma das ligações diretas de *i* para todos os outros *(g-1)* atores. Esta medida pode ser denotada pela Fórmula 4 onde *x* é a quantidade de ligações diretas de *i* para outros (*g-1*) nós *j*. Observe que esta fórmula pode indicar que um ator é bem conectado dentro de uma rede pequena ou simplesmente conectado a poucos atores dentro de uma grande rede. Para eliminar o efeito da variação do tamanho da rede na centralidade de grau, (Wasserman e Faust, 1994 apud Knoke e Yang) sugerem uma fórmula

normalizada (Fórmula 5). Um ator com alta centralidade, medida pelo seu grau é um ator que está em contato direto ou é adjacente a muitos outros atores da rede e ocupa uma localização central.

$$C_D(N_i) = \sum_{j=1}^{g} x_{ij}(i \neq j)$$

Fórmula 4 - Centralidade de grau

$$C'_D(N_i) = \frac{C_D(N_i)}{g-1}$$

Fórmula 5 – Centralidade de grau normalizada

A centralização de grau de grupo mede a variação em torno de uma tendência central, ou seja, mede o quanto diferem as centralidades individuais de grau dos atores de uma rede. É, portanto, uma medida da variância das centralidades dos nós, dada pela Fórmula 6, onde $C_A(N^*)$ é a maior centralidade de grau de um ator na rede. Desta forma, o numerador representa o somatório das diferenças do maior grau de centralização de um ator da rede e todos os demais (g-1) atores da rede e o denominador representa a soma máxima possível destas diferenças.

$$C_A = \frac{\sum_{i=1}^{g}\left[C_A(N^*) - C_A(N_i)\right]}{\max \sum_{i=1}^{g}\left[C_A(N^*) - C_A(N_i)\right]}$$

Fórmula 6 – Centralização de grau por grupo

(2) **Centralidade de proximidade**: O segundo tipo de centralidade é baseado em proximidade e distância e indica quão próximo um ator está dos outros atores em uma rede. Quanto mais central um ator, mais rapidamente ele pode interagir com os outros atores. Este tipo de centralidade depende não somente das relações diretas como também das indiretas e é calculada pela função representada pela Fórmula 7, onde o índice de centralidade de proximidade é o inverso da soma das distâncias geodésicas *d* entre o ator *i* e os demais (*g*-1) atores da rede. Para controlar o tamanho da rede e permitir comparações de atores em diferentes redes, (Beauchamp, 1965 apud Knone e Yang, 2008) propõe a função da Fórmula 8.

$$C_C(N_i) = \frac{1}{[\sum_{j=1}^{g} d(N_i, N_j)]} (i \neq j)$$

Fórmula 7 – Centralidade de proximidade

$$C'_C(N_i) = (g-1)(C_C(N_i))$$

Fórmula 8 – Centralidade de proximidade normalizada

O índice de centralização de proximidade por grupo (Fórmula 9), assim como o índice de centralização de grau por grupo, é uma medida de dispersão que mede a extensão que os atores diferem em suas centralidade de proximidade. Na fórmula, o numerador representa o somatório das diferenças da centralidade do ator com maior centralidade ($C'_C(N^*)$) e os demais atores.

$$C_C = \frac{\sum_{i=1}^{g} [C'_C(N^*) - C'_C(N_i)]}{[(g-2)(g-1)/(2g-3)]}$$

Fórmula 9 – Centralização de proximidade de grupo

(3) **Centralidade de intermediação**: Knoke e Yang (2000) exemplificam este conceito da seguinte forma: suponha que um ator *j* tem que passar por um ator *i* para se comunicar com um ator *k* (nesta situação, *i* está no caminho geodésico – menor caminho – de *j* para *k*). O ator *i* tem, então, controle sobre interações entre *j* e *k*. A quantificação da centralidade de intermediação de um ator pode ser feita com a Fórmula 10. Da mesma forma que os índices anteriores (Wasserman e Faust apud Knoke e Yang) propõem uma fórmula normalizada, onde o índice é dividido pelo valor máximo teórico de (g-1)(g-2) /2 , assumindo que cada par tem somente uma geodésica (Fórmula 11).

$$C_B(N_i) = \sum_{j<k} \frac{g_{jk}(N_i)}{g_{jk}}$$

Fórmula 10 – Centralidade de intermediação

$$C'_B(N_i) = \frac{C_B(N_i).2}{(g-1)(g-2)}$$

Fórmula 11 – Centralidade de intermediação normalizada

A centralização de intermediação em nível de grupo (Fórmula 12) mede a variância deste índice entre os atores de uma rede.

$$C_B = \frac{\sum_{i=1}^{g}[C_B(N^*) - C_B(N_i)]}{((g-1)^2(g-2))/2}$$

Fórmula 12 – Centralização de intermediação de grupo

Outras medidas de centralidade encontradas na literatura são: (i) **Centralidade de informação:** enquanto a centralidade de intermediação foca no caminho geodésico (mais curto), esta medida considera todos os caminhos no seu cálculo, focando em toda informação contida em todos os caminhos que originam num determinado ator (Wasserman e Faust, 1994); (ii) **Centralidade de autovetor**: nesta medida, a importância de um ator depende da importância de seus vizinhos (Wasserman e Faust, 1994). Um ator é mais central de acordo com a centralidade de autovetor se ele está ligado a atores que estão ligados a muitos outros, que por sua vez estão ligados a outros. O método *PageRank* (Page, et al., 1999), utilizado pelo Google na ordenação das páginas apresentadas nas busca é uma variação deste método (Okamoto, et al., 2008).

B. Prestígio

Prestígio é um outro conceito que define a importância de um ator. Basicamente o prestígio indica o quanto um ator é o receptor (observe que tal métrica só pode ser obtida onde as relações são direcionadas) em relações dentro de uma rede. Os atores de alto prestígio tendem a receber muitas nominações ou escolhas. Wasserman e Faust (1994) propõem que o grau de prestígio de um ator seja calculado como o grau de entrada de um dígrafo. A Fórmula 13 pode ser utilizada para o cálculo do prestígio. Para comparação do prestígio entre redes diferentes, utiliza-se a Fórmula 14.

$$P_D(N_i) = \sum_{j=1}^{g} x_{ji}(i \neq j)$$

Fórmula 13 – Prestígio

$$P'_D(N_i) = \frac{\sum_{j=1}^{g} x_{ji}}{g-1}(i \neq j)$$

Fórmula 14 – Prestígio normalizado

C. Componentes e cliques

Um das tarefas dos analistas de rede social é descobrir os subgrupos de uma rede, ou seja, pessoas ligadas por relações sociais informais em grupos coesivos que possuem suas próprias normas, valores, orientações e cultura que pode ir contra a estrutura social formal. Estes subgrupos ou subgrafos são conjuntos de nós de uma rede, selecionados de acordo com uma característica. O mais simples subgrafo é o componente, que é definido como o maior conjunto de nós de um grafo de acordo com determinada característica (o maior grafo que pode ser formado sem que esta característica desapareça), todos conectados entre si por caminhos (ou seja, qualquer nó pode alcançar outro nó do componente através de um ou mais caminhos) e que não possuem conexão com nós fora do componente. (John Scott,2000)

Outro conceito importante é o de clique: sua definição é similar a componente, mas enquanto neste último todos os nós são conectados via um caminho, no clique os nós são todos adjacentes. Segundo Marteleto (2005), a análise de cliques fornece um melhor entendimento de como a coesão beneficia membros do grupo. Os cliques podem representar uma instituição, um grupo específico e mesmo identificar a movimentação em torno de um determinado problema. O conceito de clique é muito incomum em redes sociais reais, de forma que algumas ideias foram propostas para ampliar o conceito: é o caso de *n*-clique, onde *n* é o tamanho máximo do caminho que liga os membros do clique. Portanto, um 1-clique é o conjunto de todos os pares conectados diretamente a uma distância 1. Um 2-clique é aquele onde os membros são conectados ou diretamente (a uma distância 1) ou indiretamente através de vizinhos comuns (a uma distância 2). E assim por diante.

D. Equivalência estrutural

Os conceitos vistos até agora estão relacionados com atores individuais e suas conexões. Um enfoque diferente pode ser feito pela análise da posição que um ator ocupa na rede. Esta posição é definida pelos tipos de relações que são mantidas por um tipo particular de ator. Dois atores podem ter conexões com indivíduos totalmente diferentes, mas os tipos de relações que eles possuem com estes indivíduos são similares. Scott (2000) exemplifica utilizando dois pais, que possuem relações com diferentes conjuntos de filhos, mas cujas relações são similares. Numa situação como esta, diz-se que os dois pais possuem equivalência estrutural. Numa definição mais formal, diz-se que dois atores são perfeitamente equivalentes estruturalmente se eles possuem padrões idênticos de relações enviadas e

recebidas dos outros atores. Ou seja, os atores *i* e *j* são equivalentes estruturalmente se, para todos os nós *k* na rede (*k* <> *i* e *k* <> *j*) se o ator *i* envia uma relação para o nó *k*, se e somente se o ator j envia uma relação para *k* e o nó *i* recebe uma relação do nó *k* se e somente se o nó *j* também recebe uma relação de *k* (Knoke e Yang, 2008) .

2.6 ARS e modelos computacionais

Os trabalhos relacionados a seguir mostram que redes complexas e a ARS têm contribuído para a solução de diferentes problemas específicos para a área de Ciência da Computação.

Pujol et al.(2002) propõe uma metodologia onde a reputação de um membro de uma comunidade eletrônica (como e-Bay, Firefly e Expert's Exchange) é definida de acordo com a posição desse membro dentro da rede social correspondente. Ou seja, as métricas da ARS são utilizadas como medida de reputação.

O artigo (Gaston e desJardins, 2003) utiliza vários experimentos para mostrar o impacto de estruturas de redes complexas na dinâmica de formação de times para realização de tarefas em um modelo computacional baseado em agentes (Weiss, 1999). Utilizando um modelo simples de organização baseado em agentes, tarefas são geradas e os agentes formam times para realizar as tarefas de maneira autônoma e descentralizada. Uma estrutura da rede (livre de escalas, mundos pequenos, randômicos e regular) é usada para restringir quais agentes podem participar em quais times. Os resultados comprovam o impacto da estrutura da rede na habilidade da organização na formação de times. Na estratégia utilizada de seleção de tarefas, a rede livre de escalas foi a mais eficiente. Posteriormente, Gaston, et al. (2004) utilizam quatro estruturas de rede para modelar diferentes topologias de interação entre os agentes (cada agente está situado em um vértice do grafo que representa a rede): aleatória, livre de escalas, reticulado regular e mundos pequenos. A definição de um time válido, que possui as habilidades coletivas necessárias para realizar uma determinada tarefa, induz um subgrafo conectado. Os resultados dos experimentos mostram também que a estrutura livre de escalas induz a maior eficiência na formação de times.

O artigo (Gaston e desJardins, 2005) define AON (*Agent-Organized Structure*) como uma estrutura organizacional da rede resultante das decisões individuais de re-conexão local feitas pelos indivíduos. O objetivo de cada indivíduo é aumentar a performance coletiva da organização: um time válido é formado por agentes com habilidades necessárias para executar

uma tarefa e, para um agente estar num time, deve ter uma conexão social com pelo menos outro agente do time. Os autores propõem estratégias para tentar desenvolver mecanismos de adaptação da rede, de forma a descobrir estruturas que rendam um melhor desempenho. Nos experimentos, quando o desempenho de um agente está abaixo da média dos vizinhos imediatos, o agente tenta adaptar a estrutura da rede. Duas estratégias de adaptação são testadas: (i) baseada na estrutura – o agente adapta sua conectividade na rede através da anexação preferencial. Em cada iteração, um agente pode escolher adaptar sua conectividade e, neste caso, seleciona randomicamente um vizinho do qual se desconectar. O agente, então, solicita uma lista de vizinhos de cada um dos seus vizinhos e seleciona dentre eles, através da anexação preferencial, o agente ao qual vai se conectar; (ii) baseada na performance - nesta estratégia, cada agente mantém uma medida de desempenho. Se seu desempenho está abaixo da média de seus vizinhos imediatos, o agente decide adaptar a estrutura da rede. Esta adaptação é baseada em desempenho e referências: primeiro, um agente a_i remove a conexão com seu vizinho imediato com o menor desempenho; o agente então solicita uma referência do vizinho com o maior desempenho. Seja a_l o agente que fornece a referência para a_i então, a_i vai estabelecer uma nova conexão com a_k, o vizinho de a_l com melhor desempenho.

Mais recentemente, em (Shinoda, et al., 2007), os autores estudam a geração de uma rede complexa onde cada agente é um nó e a adição de nova aresta é feita através de acordo entre os agentes: para cada provável aresta, os agentes votam de acordo com o aumento na sua centralidade (a medida de centralidade de um agente é utilizada como sua função de utilidade) que a aresta vai proporcionar. A aresta selecionada será aquela com maior quantidade de votos e esta nova aresta vai aumentar a utilidade de alguns agentes. O trabalho mostra que, dependendo da medida de centralidade utilizada, as redes resultantes diferem consideravelmente: a centralidade de proximidade gera uma rede livre de escalas, a centralidade de grau produz um grafo randômico, a centralidade de intermediação produz um grafo regular e a centralidade de autovetor (PageRank) produz um grafo completo.

2.7 Considerações finais

Este capítulo apresentou conceitos de aprendizagem de máquina, d*rift detection,* aprendizagem utilizando conjunto de classificadores e análise de redes sociais necessárias para o desenvolvimento do trabalho. A seguir será descrita a metodologia para a construção e validação do modelo proposto.

3 CONJUNTO DE CLASSIFICADORES SOCIAIS (CCS)

Nos capítulos anteriores foram apresentadas várias metodologias que utilizam conjuntos de classificadores em ambientes com mudança de conceito. Foi apresentada também a teoria de ARS, envolvendo a representação em grafos, os modelos de redes complexas e os conceitos da ARS. Neste capítulo será detalhado um método que aplica a Análise de redes sociais na construção e integração de um conjunto dinâmico de classificadores aplicado em um ambiente com *concept drift*. Os modelos de construção e adaptação de rede são utilizados para inserir e remover classificadores do conjunto, que formam os indivíduos da rede. Essa abordagem reduz consideravelmente a dependência de métricas locais de performance comumente observada em modelos de combinação de classificadores.

3.1 Visão geral

A metodologia propõe uma "estratégia social" para construção de um conjunto de classificadores e para integração destes classificadores. De acordo com esta estratégia, uma rede de relacionamentos é criada com os classificadores do conjunto e os conceitos da ARS são aplicados sobre esta rede. Os conceitos da ARS serão utilizados da seguinte forma:

- Durante a construção do conjunto, os conceitos serão utilizados para definir o momento em que um classificador perde sua relevância dentro do conjunto e pode ser removido;

- Na integração dos classificadores para obtenção de uma decisão coletiva, os conceitos serão utilizados para definir o peso do classificador na decisão do conjunto.

O modelo pode ser dividido em duas fases:

1. **Fase de treinamento**: etapa em que o conjunto de classificadores é construído. É durante esta fase que a rede de relacionamentos com os classificadores também é construída. Nesta rede os atores representam os classificadores e suas relações representam as afinidades (classificações iguais para os exemplos de treinamento).

Uma visão geral do método, durante a fase de treinamento, é apresentada na Figura 9 e pode ser descrita da seguinte forma:

a. Ao receber um novo exemplo de treinamento, o algoritmo verifica as predições individuais de cada classificador para o exemplo.

b. A estrutura da rede de relacionamentos é atualizada de acordo com a afinidade entre os classificadores na tentativa de melhorar o desempenho do conjunto de classificadores.

c. Utilizando a centralidade de um ator como uma medida da importância deste ator dentro da rede, uma verificação é feita para remoção de classificadores correspondentes a atores que possuam a centralidade abaixo de um *threshold*. O objetivo é excluir do conjunto os classificadores que perderam a importância nas decisões coletivas.

d. Uma vez atualizada a rede, o analisador social obtém a classificação global, ou seja, a classificação do conjunto de classificadores. Esta classificação é obtida através de votação ponderada, onde o peso do voto de cada classificador depende da centralidade de seu ator correspondente dentro da rede. Se esta classificação é diferente da classe do exemplo, ou seja, o conjunto errou ao classificar um exemplo, um novo classificador é criado, assim como seu correspondente ator, que é incluído na rede.

e. No final da iteração o novo exemplo é passado para treinamento de todos os classificadores do conjunto.

2. **Fase de classificação**: etapa em que o conjunto de classificadores é utilizado para classificar novos exemplos. O diagrama de atividades do sistema na fase de classificação é representado na Figura 10: ao receber um novo exemplo para classificação, o algoritmo verifica as predições individuais dos classificadores, obtém a predição global (conforme visto anteriormente, os classificadores utilizam as centralidades de seus atores na rede criada na fase de treinamento para a decisão social) utilizando a rede de relacionamentos criada na fase de treinamento e retorna a predição coletiva.

3.2 Desenvolvimento

Para a implementação da metodologia proposta, alguns aspectos devem ser definidos. Um deles é o modelo a ser utilizado para construção da rede de relacionamentos. Como visto na seção 2.5.3, diversos modelos foram propostos na tentativa de reproduzir os modelos de rede do mundo real. No caso de um conjunto de classificadores, onde os membros possuem reputações diferentes, as ligações entre os membros são aleatórias, e a quantidade de indivíduos pode variar em função das escolhas coletivas realizadas, o modelo de rede escolhido foi o modelo livre de escalas.

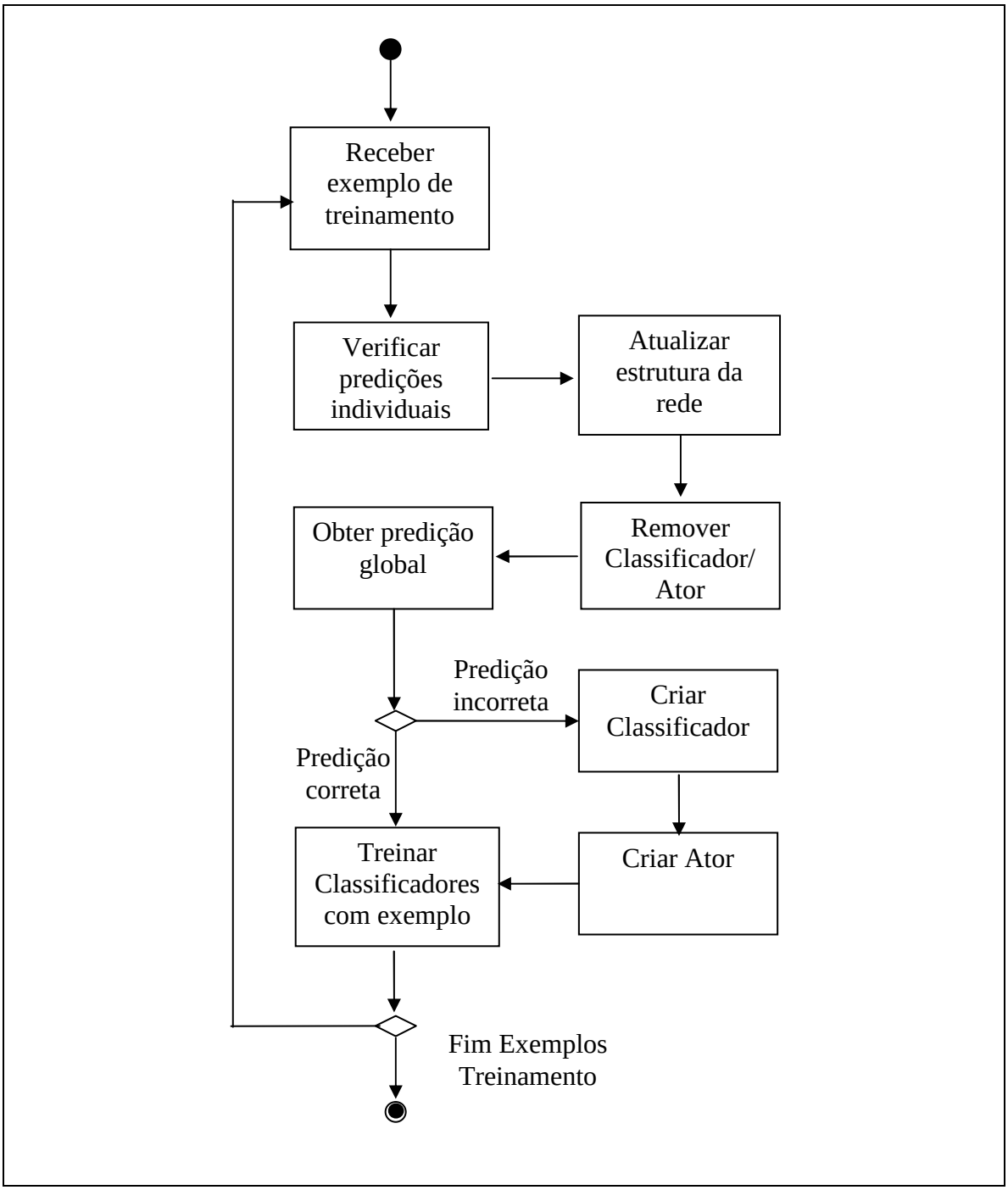


Figura 9 – Fluxo de atividades -módulo de treinamento

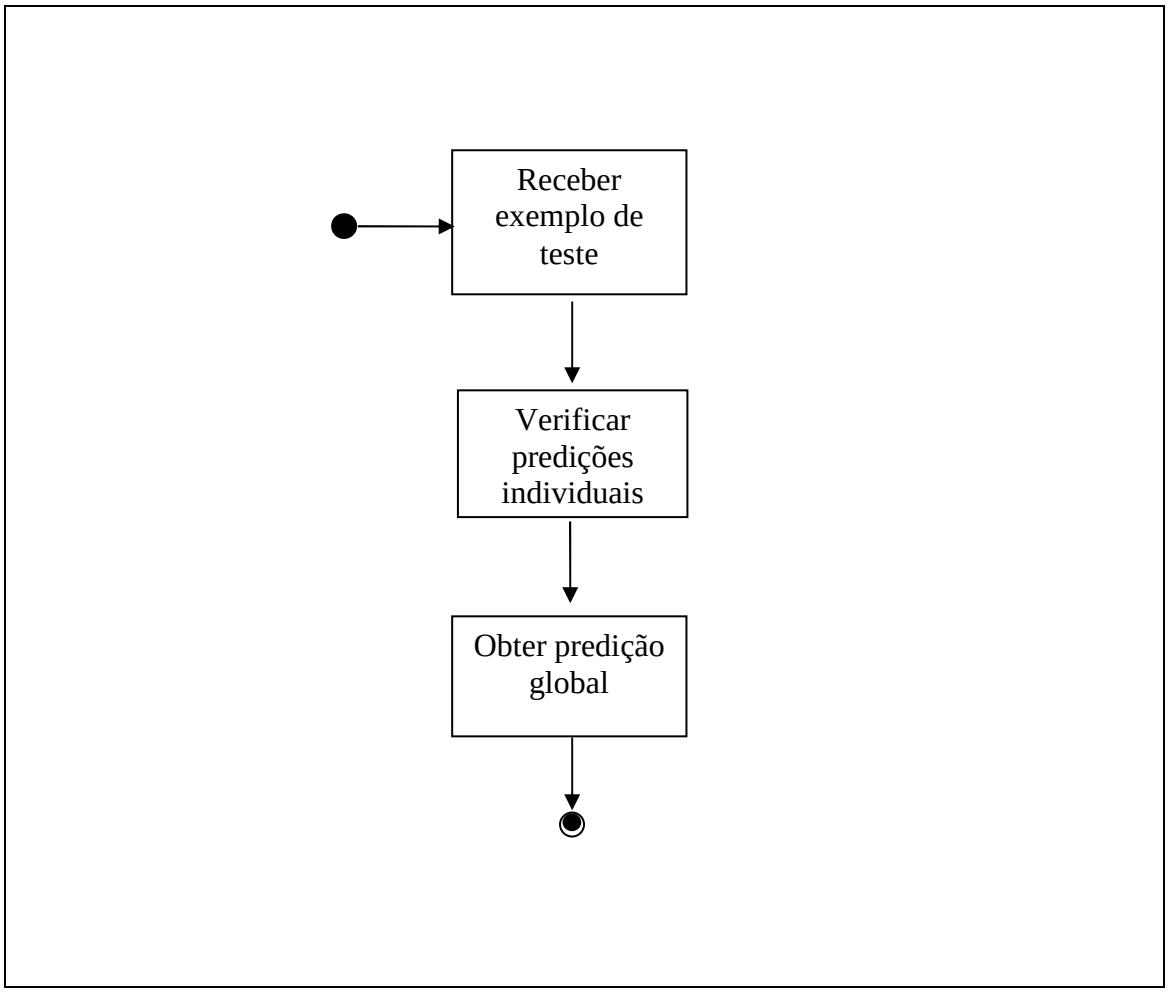


Figura 10 – Fluxo de atividades - Módulo de classificação

Uma outra opção foi a utilização de implementações de algoritmos da biblioteca Java WEKA (*Waikato Environment for Knowledge Analysis*) (Witten e Frank, 2005) mais especificamente o IBK, que implementa o algoritmo K-NN e o J48, baseado no algoritmo C4.5. Esta biblioteca[6] compreende um grande número de implementações em Java de algoritmos de aprendizagem para tarefas de mineração de dados como árvores de decisão, máquinas de vetor de suporte, classificadores baseados em instâncias, redes neurais, redes Bayesianas e *clustering*. Além disto, fornece meta classificadores como *bagging* e *boosting*, métodos de avaliação como validação cruzada e *bootstraping*, inúmeros métodos de seleção de atributos e técnicas de pré-processamento. O *software* possui uma interface gráfica que permite a carga de dados, a aplicação de algoritmos de aprendizagem e a visualização dos modelos construídos e uma interface Java disponível para todos os algoritmos, que permite integrá-los em qualquer programa (Dimov, 2007). Esta facilidade de integração faz com que a

[6] Disponível em http://www.cs.waikato.ac.nz/~ml/weka/

mudança do algoritmo base de aprendizagem dos classificadores seja rapidamente implementada.

3.2.1 O Algoritmo CCS

O algoritmo CCS pode ser resumido no pseudo-código apresentado na **Erro! A origem da referência não foi encontrada.**. O algoritmo trabalha de forma iterativa, recebendo um exemplo a cada passo do treinamento. Seu detalhamento será feito utilizando um cenário de aprendizagem composto por uma série de exemplos classificados *S*, sendo $S = \{x_t, y_t\}$, para $t = 1, 2, 3,.....T$, onde x_t é o vetor de restrições (ou características) e y_t a classe. A série *S* é apresentada ao sistema e, a cada exemplo, o algoritmo gera uma saída Λ_t, que é a classificação global do conjunto de classificadores para um exemplo *t*.

A cada passo, ou seja, a cada exemplo $\{x_t, y_t\}$ processado, o procedimento do algoritmo é o seguinte:

1. verifica as predições individuais dos classificadores e duas listas são criadas com os atores correspondentes aos classificadores: uma, com os atores com predição correta e outra com os atores com predição incorreta.

2. A seguir, a rede de relacionamentos é atualizada (procedimento Atualiza-Relacoes apresentando na **Erro! A origem da referência não foi encontrada.**). Esta atualização ocorre da seguinte forma: o ator incorreto com a menor centralidade de grau (quando houver mais de um ator, seleciona um deles aleatoriamente) tenta substituir uma relação existente com outro ator incorreto, selecionado através de uma probabilidade inversamente proporcional à centralidade do ator na rede, por um relacionamento com um ator correto, sendo que este último é selecionado de acordo com uma probabilidade proporcional à centralidade. O relacionamento com o ator incorreto é removido e o novo relacionamento é criado, se este relacionamento ainda não existir. Este processo de adaptação é realizado para aumentar o desempenho do conjunto e foi baseado nos estudos de Gaston e deJardins (2005). Tais estudos utilizam a estrutura de redes complexas na formação de times de agentes, e mostram que a eficiência organizacional pode ser melhorada quando os agentes podem adaptar suas relações locais dentro de uma rede (seção 2.5).

3. Continuando o processo de adaptação da rede, é realizada uma verificação para remoção de atores com baixa reputação (procedimento Remove-Classificadores - **Erro! A origem da referência não foi encontrada.**): Dentre todos os atores com centralidade abaixo

de um *threshold* e cuja predição individual para o exemplo corrente está incorreta, o ator com menor centralidade é removido[7].

CCS ($\{\vec{x}, y\}, c, \theta, m$)

$\{\vec{x}, y\}$	: dados de treinamento: $\vec{x}$ = vetor de características e y = classe
$c \in \mathrm{N}^*$	: classes, $c \geq 2$
θ	: *threshold* para remoção de classificadores
$\{e,\lambda\}\ \lambda \in \{1,..,c\}$	: conjunto de classificadores e suas predições individuais
$\Lambda \in \{1,..,c\}$	: predição global do conjunto
$\vec{\sigma} \in \mathrm{N}^*$	: soma das predições ponderadas para cada classe
$G(A,L)$	: grafo que representa uma rede de relacionamentos onde $A=\{a_1, ..., a_c\}$ é o conjunto de atores e $L=\{l_1,...,l_z\}$ é o conjunto de arestas
$l(a_i, a_j)$	: uma aresta não direcionada conectando os atores a_i e a_j;
m	: quantidade inicial de relacionamentos de um novo ator dentro da rede
a_i	: um ator i
$C_G(a_i)$	: centralidade de grau do ator a_i

1. Para $i=1$ até *numExemplosTreino Faça* {
2. *Corretos = Incorretos = Ø*
3. Para $j=1$ até *numClassificadores Faça* {
4. λ_j = *Classifica*(e_j, $\vec{x}_i$)
5. Se $\lambda_j = y_i$, *Corretos = Corretos* $\cup$ { a_j }
6. Senão, *Incorretos = Incorretos* $\cup$ { a_j }
7. }
8. *Atualiza-Relacoes (Corretos, Incorretos, G)*
9. *Remove-Classificadores(Incorretos,{e},G)*
10. Λ = *Obtem-Predicao-Social({e,λ},G)*
11. Se ($\Lambda \neq y_i$) *ou (numClassificadores=0){*
12. adiciona novo classificador e_N em *{e}*
13. adiciona novo ator a_N em *A*
14. *Cria-Relacoes-Iniciais(a_N,G,m)*
15. }
16. Para $j=1$ até *numClassificadores F*aça{
17. e_j = *Treina*(e_j, $\{\vec{x}, y\}_i$)
18. }
19. Saída: Λ
20. }
21.
22. Fim

Figura 11 – Algoritmo Conjunto de Classificadores Sociais (CCS)

[7] Na implementação, quando mais de um ator se encontra na mesma situação para remoção, o mais antigo é removido.

Atualiza-Relacoes(CORRETOS, INCORRETOS,G)

1. α_{mC} = *Ator-Menor-Centralidade(INCORRETOS)*
2. *REMOVIDOS = Ø*
3. Para cada ator $\alpha_i \in$ *INCORRETOS* faça {
4. Se $l(a_{mC}, a_{i)} \in L$, então *REMOVIDOS = REMOVIDOS* $\bigcup$ { α_i }
5. }
6. Π_{inv} = *Probabilidade Inversamente proporcional a centralidade de grau de cada ator em REMOVIDOS*
7. α_{rem} = Seleciona ator em *REMOVIDOS* para remover relacionamento com α_{mC} usando Π_{inv}
8. $L \leftarrow$ Remove-Relacao(α_{rem}, α_{mC})
9. Π = *Probabilidade proporcional a centralidade de grau de cada ator em CORRETOS*
10. α = Seleciona ator em *CORRETOS* para criar relacionamento com α_{mC} usando Π
11. Se $l(a, a_{mC)} \notin L$, então $L \leftarrow L \bigcup \{ l(a, a_{mC)} \}$

Figura 12 – Procedimento Atualiza-Relacoes

Remove-Classificadores *(INCORRETOS, {e},G, θ)*

1. C_{Gmaior} = maior centralidade dentre as centralidades dos atores de *A*
2. α_{menor} = Seleciona ator em *INCORRETOS* com menor centralidade
3. Se $C_G(\alpha_{menor}) < \theta \times C_{Gmaior}$ {
4. Remove ator α_{menor} de *A*
5. Remove classificador *e* correspondente a α_{menor} de *{e}*
6. }

Figura 13 – Procedimento Remove-Classificadores

4. Após a atualização da rede, a classificação do exemplo pelo conjunto é obtida da seguinte forma: uma votação é realizada, e cada voto é ponderado pela centralidade do ator correspondente (procedimento Obtem-Predicao-Social - Figura 14). A classificação do conjunto corresponde à classe com maior votação[8].

5. O próximo passo é verificar a inclusão de um novo classificador no conjunto. A inclusão é feita quando a predição do conjunto for incorreta. Nesta situação, um novo classificador é criado, assim como seu ator correspondente. O novo ator é incluído na rede de relacionamento utilizando a anexação preferencial: serão criados *m* relacionamentos para o ator, sendo que o ator adjacente é selecionado através da anexação preferência (procedimento Cria-Relacoes-Iniciais - Figura 15).

Obtem-Predicao-Social(*{e,λ},G)*

1. $\vec{\sigma} \leftarrow 0$
2. Para i = 1 até *numClassificadores* Faça {
3. $\alpha_i \leftarrow$ ator correspondente ao classificador e_i
4. $\sigma_\lambda = \sigma_\lambda + C_G(\alpha_i)$
5. }
6. Saída: arg $\max_i \sigma_i$

Figura 14 – Procedimento Obtem-Predicao-Social

6. No final, todos os classificadores são treinados com o exemplo corrente.

Cria-Relacoes-Iniciais(*a, G, m*)

1. i = 0
2. Enquanto $i < m$ Faça {
3. Π = Probabilidade do ator *a* em *A*, proporcional a sua centralidade, de
4. receber uma nova ligação
5. Selecione ator a_{sel} em *A* de acordo com Π
6. $L \leftarrow L \cup l(a, a_{sel})$
7. $i = i + 1$
8. }

Figura 15 – Procedimento Cria-Relacoes-Iniciais

3.3 Análise preliminar

A metodologia CCS se propõe a explorar uma rede de relacionamentos, cujos atores representam os classificadores do conjunto, onde alguns atores se tornam mais relevantes à medida que a rede evolui. Tratando-se de um modelo livre de escalas, espera-se que durante esta evolução alguns atores atraiam mais relacionamentos, tornando-se *hubs* dentro da rede. Estes atores com maior relevância na rede teriam o peso maior na decisão do conjunto.

[8] Na implementação, quando ocorre um empate entre as classes, a classe selecionada é a primeira classe definida (ver no Anexo I um exemplo de arquivo no formato .ARFF (padrão WEKA), onde o atributo classe é o último atributo definido)

Para efeito de ilustração, uma avaliação simplificada da metodologia será realizada para observar a evolução da rede de classificadores. Serão utilizados o algoritmo 1-NN para facilitar o entendimento e um exemplo de domínio baseado em (Stanley,2003), onde são modeladas as preferências automobilísticas de um consumidor jovem.

Considerando os exemplos de treinamento da Tabela 4, que representam as escolhas feitas por consumidores com este perfil, serão apresentadas as configurações do conjunto de classificadores e da rede de relacionamento a cada iteração.

Exemplo	**Tamanho**	**Cor**	**Marca**	**Classe**
e_1	Pequeno	Azul	Toyota	Não interessado
e_2	Pequeno	Verde	Toyota	Interessado
e_3	Grande	Verde	Honda	Não interessado
e_4	Grande	Azul	Honda	Não interessado
e_5	Pequeno	Verde	Honda	Interessado

Tabela 4 – Exemplos de escolhas automobilísticas (baseado em Stanley, 2003)

Na primeira iteração, um classificador c_1 e seu correspondente ator $a1$ são criados e c_1 é treinado com o exemplo e_1 (Figura 16).

1ª Iteração	$c_1\{ e_1\}$	

Figura 16 – Configuração do conjunto de classificadores e rede de relacionamentos (1ª Iteração)

Na segunda iteração, é verificada a predição individual do classificador para o exemplo e_2, que neste caso é a mesma predição do conjunto. Como o vizinho mais próximo é e_1, erroneamente c_1 prediz e_2 como "Não interessado". O erro na predição do conjunto faz com que novo classificador c_2 seja criado, assim como seu correspondente ator a_2. Os classificadores c_1 e c_2 são treinados com o exemplo (Figura 17).

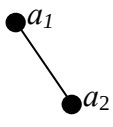
2ª Iteração	$c_1\{ e_1, e_2\}$ $c_2\{ e_2\}$	

Figura 17 - Configuração do conjunto de classificadores e rede de relacionamentos (2ª Iteração)

Na terceira iteração, ilustrada na Figura 18, c_1 prediz o exemplo e_3 incorretamente enquanto c_2 prediz corretamente. Se considerarmos que o voto c_1 prevalece c_2, a predição global será incorreta. Nesta situação, um novo classificador c_3 e um novo ator a_3 serão criados. O ator será incluído na rede e terá suas conexões iniciais criadas de acordo com a anexação preferencial, ou seja, os atores com maior centralidade terão maior probabilidade de receber esta ligação de a_3. Como nesta situação a rede possui somente dois atores, e considerando que m (quantidade de ligações iniciais) seja igual a 1, a probabilidade de que a_1 e a_2 recebam ligação de a_3 é de 50% cada. Os classificadores do conjunto são treinados com e_1, e_2 e e_3.

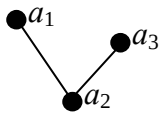
3ª Iteração	c_1\{ e_1, e_2, e_3\} c_2\{ e_2, e_3\} c_3\{ e_3\}	

Figura 18- Configuração do conjunto de classificadores e da rede de relacionamentos (3ª Iteração)

Na quarta iteração (Figura 19), os três classificadores acertam a predição e todos eles são treinados com e_4. A rede não sofre alteração.

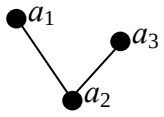
4ª Iteração	c_1\{ e_1, e_2, e_3, e_4\} c_2\{ e_2, e_3, e_4\} c_3\{ e_4\}	

Figura 19 - Configuração do conjunto de classificadores e da rede de relacionamentos (4ª Iteração)

Na quinta iteração (Figura 20), c_2 e c_3 têm a predição individual incorreta enquanto c_1 possui predição correta. Como a centralidade de α_2 é maior, seu voto terá maior peso na decisão. Portanto, a predição do conjunto será incorreta e um novo classificador c_4 será criado. Nesta situação, a probabilidade de que a_2 receba a ligação de a_4 será maior que a_1 e a_3.

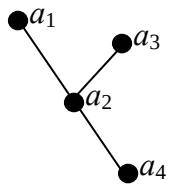
5ª Iteração	c_1\{ e_1, e_2, e_3, e_4, e_5\} c_2\{ e_2, e_3, e_4, e_5\} c_3\{ e_4, e_5\} c_4\{ e_5\}	

Figura 20 - Configuração do conjunto de classificadores e da rede de relacionamentos (5ª Iteração)

Observar que durante estas iterações existe uma adaptação dos relacionamentos entre os atores para tentar melhorar o seu desempenho, que não foi ilustrada. Esta adaptação consiste em fazer com que um ator, utilizando certos critérios associados ao desempenho individual e a estrutura da rede, se desligue de um ator e se ligar a outro com o qual não possua relacionamento. Além da adaptação, acontece a remoção de classificadores com pequena relevância (baixa centralidade) dentro da rede.

A expectativa é que durante a fase estável de um conceito a rede se estabilize com os mesmos classificadores. Porém quando ocorre uma mudança de conceito, os classificadores do conjunto devem cometer mais erros de predição e, consequentemente, devem perder relacionamentos, diminuindo sua importância dentro da rede, e até mesmo podem ser removidos, sendo substituídos por novos classificadores.

Com esta metodologia espera-se que os melhores classificadores sejam mantidos enquanto os piores sejam descartados, garantindo um bom desempenho do conjunto na parte estável dos dados e uma rápida recuperação do desempenho após uma mudança de conceito.

3.4 Considerações finais

Neste capítulo foi apresentado o algoritmo CCS. A seguir serão apresentados: a metodologia de avaliação, os experimentos preliminares necessários para definição da metodologia e os resultados obtidos.

4 RESULTADOS OBTIDOS

Neste capítulo serão apresentados a metodologia de avaliação, os experimentos preliminares necessários para avaliação da metodologia CCS e os resultados obtidos na aplicação desta metodologia em ambientes com mudança abrupta, moderada e gradual. Estes resultados serão comparados com os resultados obtidos com o algoritmo DWM, selecionado devido ao seu desempenho quando comparado com outras metodologias e algumas similaridades com a metodologia CCS. A comparação será feita utilizando os algoritmos K-NN e J48 em ambas as metodologias.

4.1 Metodologia de avaliação

Nesta seção será feita uma explicação da base de dados utilizada para avaliar o algoritmo, bem como as medidas de desempenho que foram utilizadas para avaliação.

4.1.1 Dados de treinamento e de testes

A base de dados utilizada para avaliação do modelo proposto deve apresentar a característica de *concept drift*. A opção foi feita pela utilização do mesmo domínio de (Enembreck, et al., 2009): um processo de negociação bilateral entre um vendedor e um comprador, onde um agente (comprador) deve ser capaz de identificar mudanças na política de negociação de seu oponente (vendedor), ou seja, se uma oferta será interessante ou não para o vendedor. Para tentar identificar uma mudança na política do oponente, o agente utiliza a detecção de mudança. Neste domínio, uma oferta pode ser definida pelas características apresentadas na Tabela 5. O conceito representa a descrição de uma oferta interessante e uma mudança de conceito representa uma mudança na política de negociação do comprador. O procedimento para geração de exemplos de treinamento e teste foi baseado no sistema *Stagger* (Schlimmer e Granger, 1986). Enembreck et al. (2009) definiu alguns conceitos disjuntos que o vendedor pode usar e o comprador deve descobrir. Os exemplos são classificados como *Interessante* e *Não interessante*.

Característica	Domínio
Cor	[preta, azul, ciano, marrom, vermelha, verde, amarela, magenta]
Preço	[muito_baixo, baixo, normal, alto, muito_alto, bastante_elevado, enorme, não_vendável]
Pagamento	[0, 30, 60, 90, 120, 150, 180, 210, 240]
Quantidade	[muito_baixa, baixa, normal, alta, muito_alta, bastante_elevada, enorme, não_garantida]
Prazo de entrega	[muito_curto, curto, normal, longo, muito_longo]

Tabela 5 – Características de uma oferta em um processo de negociação bilateral

Como alguns algoritmos lidam bem com mudança de conceito abrupta, mas não tão bem com mudança de conceito moderada ou gradual, foram criadas bases de dados que simulam estas três formas de mudança de conceito. Os conceitos utilizados para treinamento e teste estão apresentados nas tabelas 6, 7 e 8.

Conceito	Descrição de uma oferta interessante	Descrição do conceito
1	(Preço=normal e Quantidade=grande) ou (Cor=marrom e Preço=muito_baixo e Prazo de entrega = longo)	Uma oferta é considerada interessante caso o preço seja normal e a quantidade desejada do produto é grande. No entanto, um problema na linha de produção de produtos da cor marrom força o vendedor a reduzir o preço porque o prazo de entrega provavelmente será longo.
2	(Preço=Alto e Quantidade=muito_grande e Prazo de entrega=muito_curto)	Os fornecedores enfrentam dificuldade para fornecer matéria-prima e a produção tem diminuído drasticamente. Por causa da falta de produtos, a companhia penaliza clientes que compram grandes quantidades e necessitam delas com urgência com um aumento para entrega nestas condições.
3	(Preço=muito_baixo e Pagamento=0 e Quantidade=grande) or (Cor=vermelha e Preço=baixo e Pagamento=30)	Clientes importantes declaram incapacidade de honrar grandes contratos e a empresa precisa de capital para continuar operando. Portanto, a empresa vai reduzir drasticamente os preços de todos os produtos para pedidos de grandes quantidades e pagamento em dinheiro. No entanto, produtos vermelhos estão em falta, o que impede que seu preço caia muito. Para compensar isso, a empresa decide prolongar o prazo de pagamento para 30 dias.
4	(Cor=preta e Pagamento=90 e Prazo de entrega=muito_curto) or (Cor=magenta e Preço=alto e Prazo de entrega=muito_curto)	A empresa mantém uma grande quantidade de produtos pretos, implicando em alto custo de armazenamento e logística. A empresa então estende o prazo de pagamento para este produto, garantindo entrega imediata. Produtos magenta agora são produzidos de matéria-prima importada, tendo um aumento no seu custo. No entanto, há também uma grande quantidade deste produto em estoque que precisa ser vendida rapidamente.
5	(Cor=azul e Pagamento=60 e Quantidade=pequena e Prazo de entrega=normal) or (Cor=ciano e Quantidade=pequena e Prazo de entrega=normal)	Grandes investidores estão agora investindo na empresa, aumentando seus recursos de capital, que começa a se posicionar de forma agressiva para ganhar quota de mercado por meio da popularização de alguns dos seus produtos. A empresa garante longos prazos de pagamento para produtos ciano e azul, mesmo para pedidos de pequenas quantidades.

Tabela 6 – Conceitos utilizados na mudança abrupta

Conceito	Descrição de uma oferta interessante
1	(Prazo de entrega = muito_baixa e Quantidade = muito_pequena)
2	(Prazo de entrega = muito_baixa e Quantidade = pequena)
3	(Prazo de entrega = muito_baixa e Quantidade = normal)
4	(Prazo de entrega = muito_baixa e Quantidade = alta)
5	(Prazo de entrega = muito_baixa e Quantidade = muito_alta)
6	(Prazo de entrega = muito_baixa e Quantidade = bastante_elevada)
7	(Prazo de entrega = muito_baixa e Quantidade = enorme)
8	(Prazo de entrega = muito_baixa e Quantidade = não_garantida)

Tabela 7 – Conceitos utilizados na mudança moderada

Conceito	Descrição de uma oferta interessante
A	(Cor=azul e Prazo_entrega=muito_curto)
B	(Cor=preta e Preço=alto) ou (Cor=magenta e Pagamento=0)

Tabela 8 – Conceitos utilizados na mudança gradual

Como visto na seção 2.3, o que diferencia as formas de mudança abrupta e gradual é o tamanho da zona de *drift* (Δx na Figura 6): no caso da mudança abrupta, $\Delta x = 1$ (somente um exemplo) e no caso da mudança gradual, $\Delta x > 1$, o que significa que o conceito está mudando durante um certo número de exemplos.

Stanley (2003) modela a mudança de conceito utilizando uma função α, que representa o domínio do conceito *A* sobre o conceito *B* em um exemplo específico. Se *A* está estável até o momento *x*, então até o exemplo i_x, $\alpha = 1$. E após $i_x + \Delta x$, $\alpha = 0$. Enquanto o conceito está mudando, $0 \leq \alpha \leq 1$. Nesta situação, a probabilidade que um exemplo esteja no conceito *A* é dado por $p(A) = \alpha$ e a probabilidade que o mesmo exemplo esteja no conceito *B* é $p(B) = 1 - \alpha$. A função α é representada na Figura 21, onde o eixo *x* representa o tempo.

Na terceira forma de mudança avaliada, a mudança moderada, assim como na abrupta, a mudança de conceito ocorre de um exemplo para outro ($\Delta x = 1$), porém um novo conceito mantém certa similaridade com o conceito anterior.

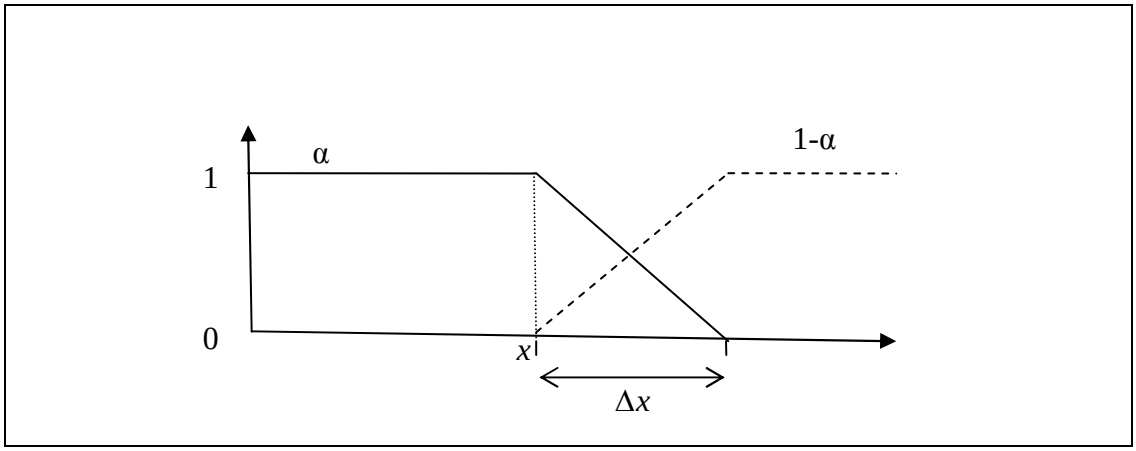


Figura 21 – A função α (baseado em Widmer e Kubat, 1996)

O procedimento de geração dos exemplos de treinamento e teste[9] são apresentados nos diagramas de atividade das figuras 22 e 23. Segundo Stanley (2003) a distribuição dos exemplos nos arquivos de teste deve refletir a distribuição da qual o conceito do exemplo corrente de treinamento foi escolhido. Isto explica a diferença na geração dos arquivos de teste nas zonas estável e de *drift*. Durante a parte estável, o arquivo de teste representa o conceito corrente, conforme o diagrama de atividades da Figura 22, cujo resultado é apresentado na Figura 24 (para cada exemplo de treinamento de um conceito, o arquivo de testes é o mesmo). Na zona de *drift*, para cada exemplo de treinamento gerado de acordo com *p(A)* e *p(B)*, deve haver um arquivo de testes com exemplos, distribuídos de acordo com *p(A)* e *p(B).* A Figura 23 mostra o diagrama de atividades para a geração do arquivo de treinamento (com Δx exemplos) e Δx arquivos de teste para a zona de *drift* entre dois conceitos A e B. O resultado é apresentado na Figura 25 (para cada exemplo de treinamento, o arquivo de testes é um arquivo diferente).

[9] Os arquivos de treinamento e testes foram criados no formato .arff (*Attribute-Relation File Format*), que é o formato esperado pelos componentes do WEKA (exemplo no Anexo I).

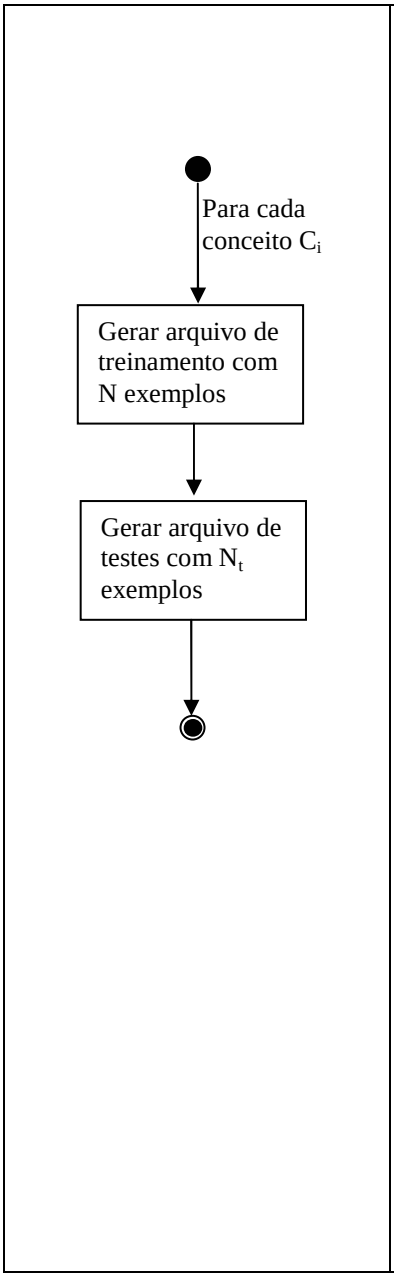


Figura 22 –Fluxo de atividades para geração de arquivos de treinamento e teste para a fase estável dos dados

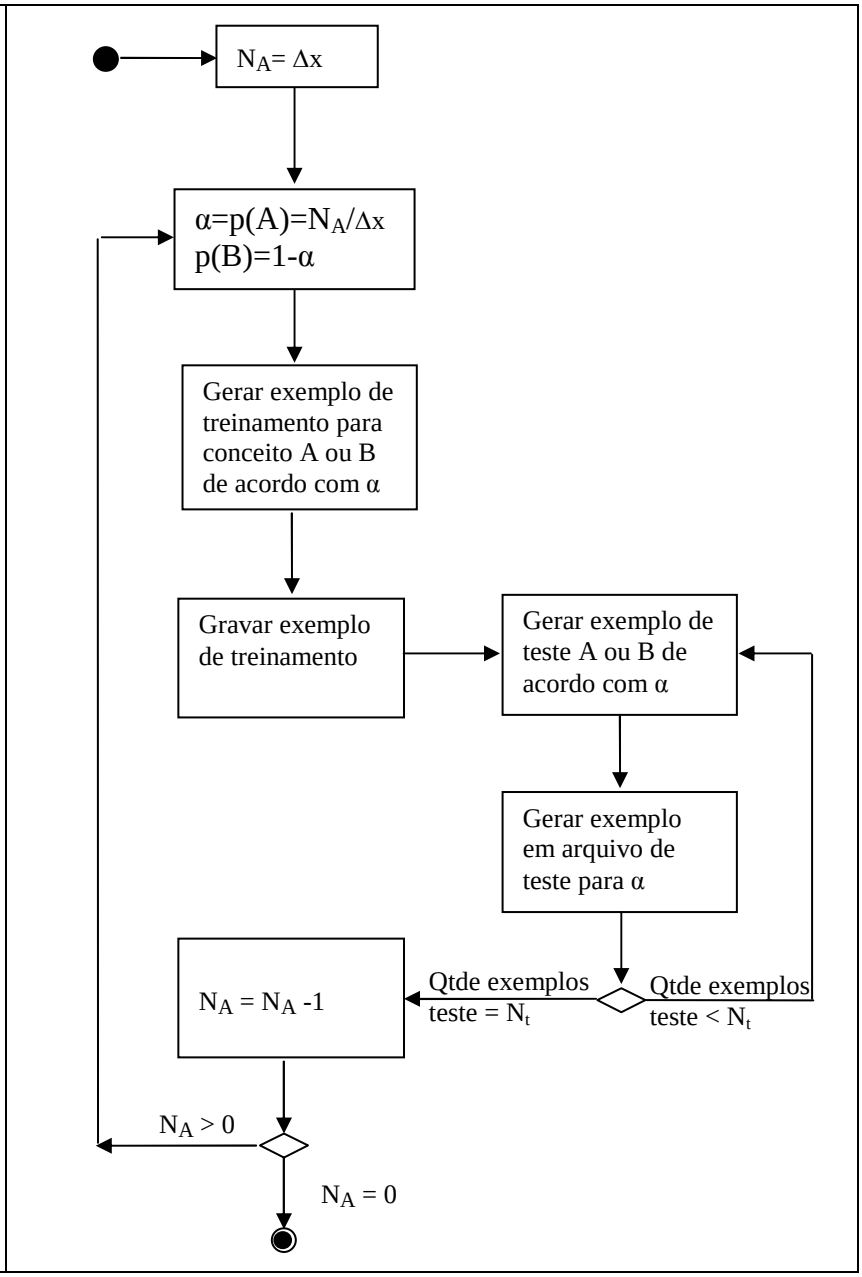


Figura 23 – Fluxo de atividades para a geração de arquivos de treinamento e teste para a zona de *drift* entre os conceitos A e B.

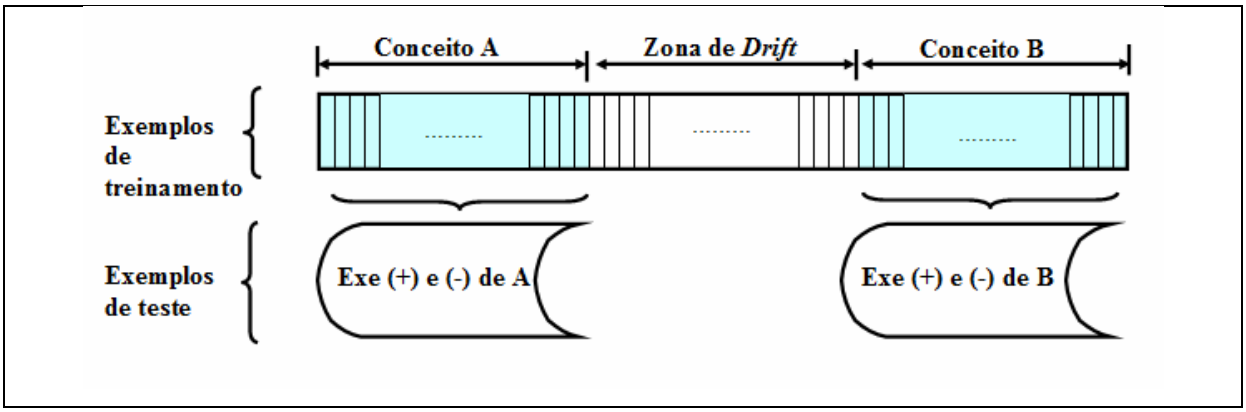


Figura 24 – Resultado do fluxo de atividades da Figura 21

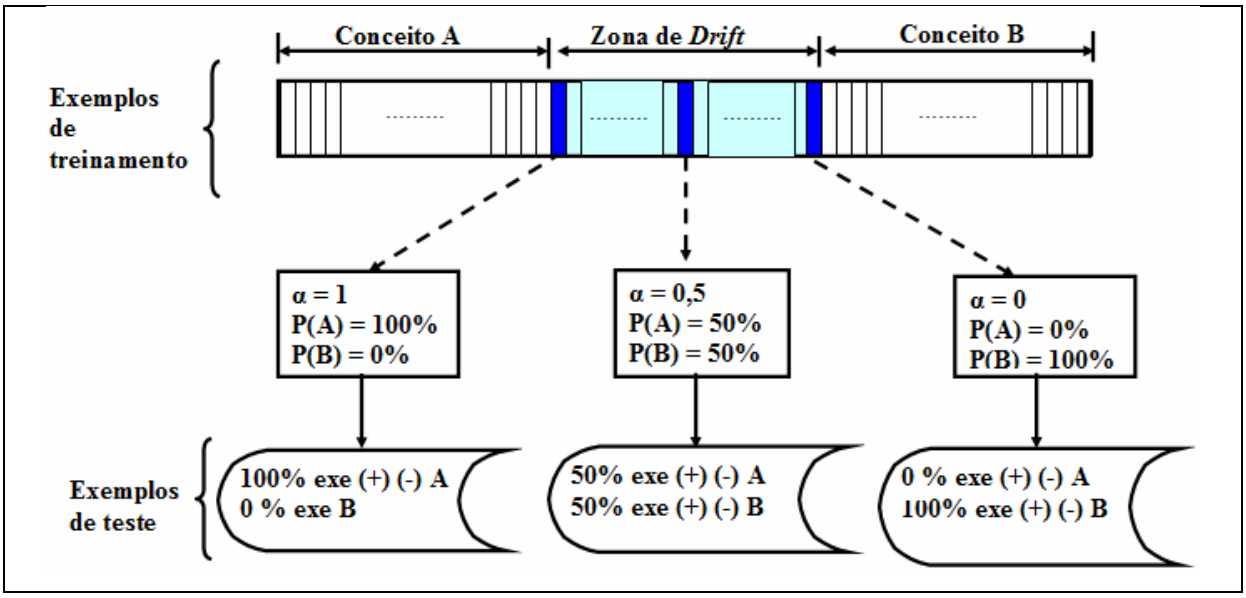


Figura 25 – Resultado do fluxo de atividades da Figura 22

4.1.2 Medidas de desempenho

A avaliação do algoritmo proposto será feita utilizando as seguintes medidas de desempenho:

1. Taxa de acerto: a cada passo da fase de treinamento, ou seja, a cada exemplo de treinamento, o comitê de classificadores que está sendo construído é avaliado através da submissão de um arquivo com exemplos de teste ao sistema. A Figura 26 ilustra o procedimento: para cada conceito *c*, um número de exemplos de treinamento é submetido ao modelo. A cada passo (*i*), o modelo recebe um exemplo de *c* e é avaliado com o arquivo de teste correspondente ao conceito. É possível observar que na zona de *drift* o arquivo de teste

vai depender da função α que rege a distribuição de exemplos no arquivo de treinamento. Como cada exemplo de treinamento da zona de *drift* foi gerado de acordo com uma probabilidade, os exemplos de teste para cada exemplo de treinamento processado deve seguir a mesma probabilidade (Figura 27).

$$Taxa\ de\ acerto(c,i) = \frac{N}{T} \times 100$$

Fórmula 15 - Precisão do sistema para predição do conceito *c* no instante *i*.

A função para avaliação da taxa de acerto é representada pela Fórmula 15, sendo *N* a quantidade de exemplos de teste classificados corretamente e *T*, o total de exemplos de teste. A recuperação da taxa de acerto (a quantidade de exemplos que o algoritmo necessita para recuperar o desempenho) após uma mudança de conceito vai indicar o quão rápido o algoritmo se adapta à esta mudança.

2. Quantidade de classificadores do conjunto: métrica avaliada a cada passo da fase de treinamento, durante a construção da rede.

3. Idade média dos classificadores no conjunto: métrica (calculada pela fórmula 16) avaliada a cada passo da fase de treinamento, durante a construção da rede. Trata-se da média da idade dos *n* classificadores que compõem o conjunto.

$$Idade\ média = \frac{\sum_{i=1}^{n} idade(i)}{n}$$

Fórmula 16 – Idade média dos classificadores de um conjunto

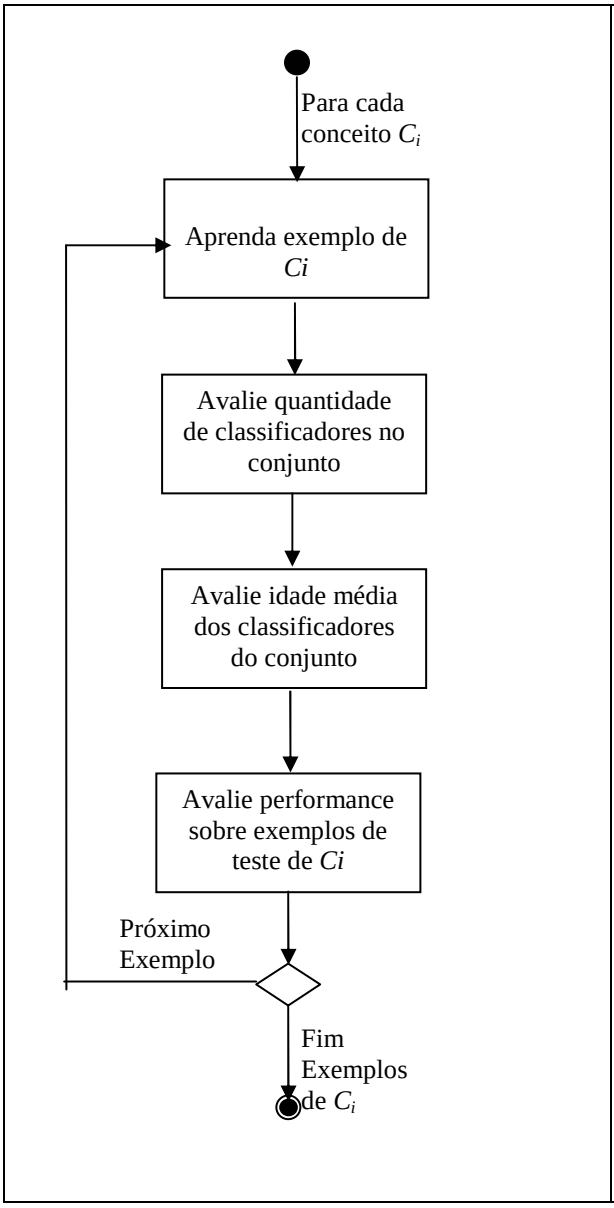


Figura 26 – Fluxo de atividades para a avaliação na zona estável

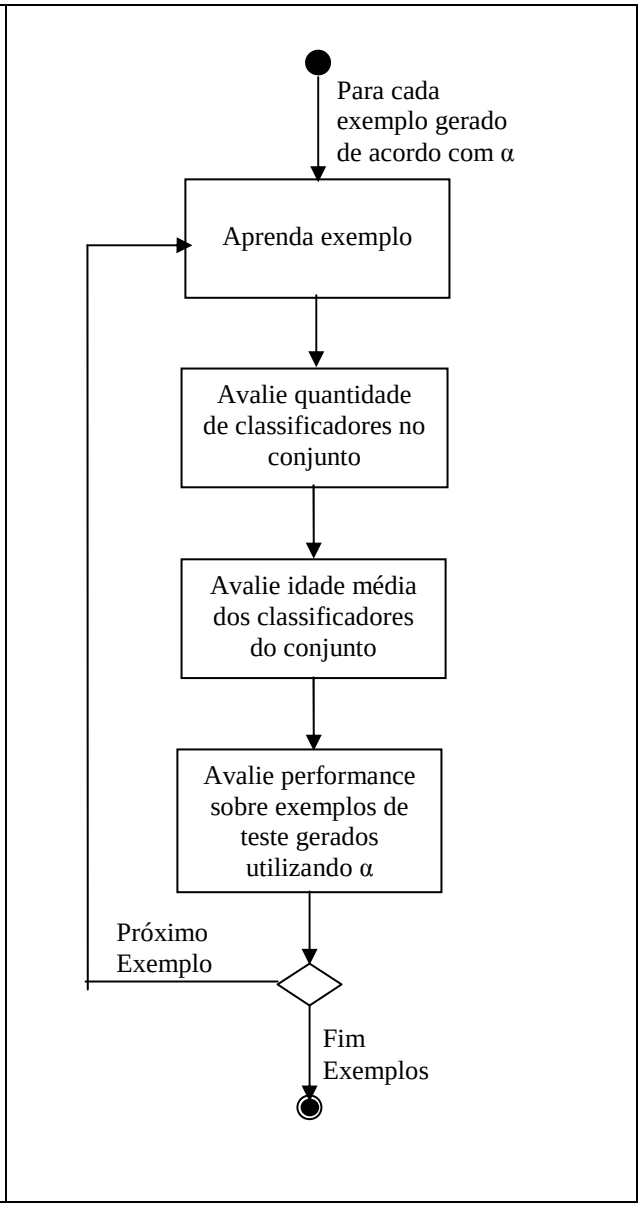


Figura 27 – Fluxo de atividades para avaliação na zona de *drift*

4.1.3 Experimentos preliminares

Algumas definições devem ser feitas antes dos experimentos. A primeira definição a ser feita é qual valor de K a ser utilizado no algoritmo K-NN. As figuras 28, 29 e 30 apresentam o desempenho da aplicação da metodologia DWM utilizando o algoritmo K-NN com vários valores de K (no Anexo II são apresentados os gráficos com a quantidade de classificadores no conjunto) e verifica-se que o desempenho num ambiente com mudança abrupta e gradual é obtido com K=1. No caso de mudança moderada, o melhor desempenho é com K=5. Em

todos os tipos de mudança, K=3 sempre ficou com a média e portanto será o algoritmo utilizado neste trabalho.

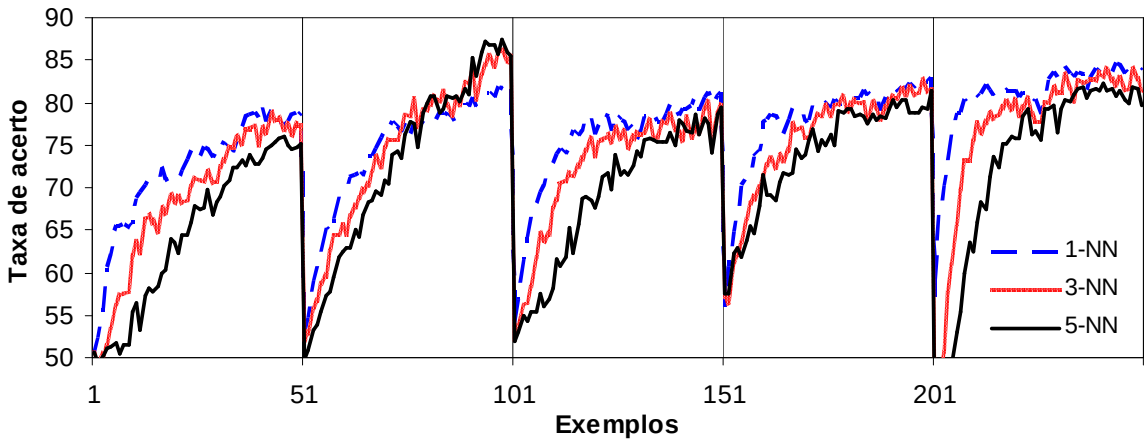


Figura 28 – Taxa de acerto - algoritmos K-NN (DWM – Mudança abrupta)

Figura 29 – Taxa de acerto - algoritmos K-NN (DWM – Mudança moderada)

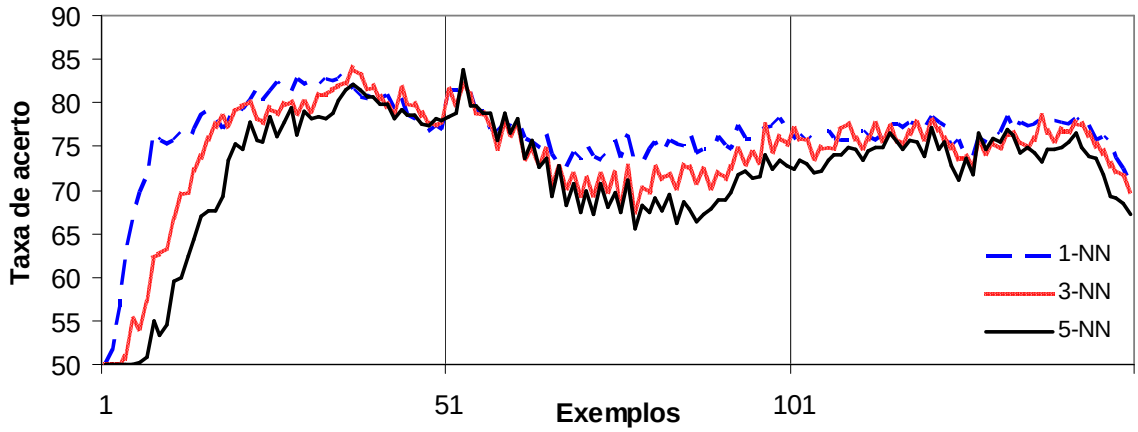


Figura 30 – Taxa de acerto - algoritmos K-NN (DWM – Mudança gradual)

Para tentar o encontrar o equilíbrio entre a quantidade de classificadores e o desempenho do conjunto (o ideal é conseguir a utilização mais eficiente dos recursos, ou seja, o melhor desempenho com a menor quantidade possível de classificadores) outros experimentos foram realizados com este *framework* para definir alguns parâmetros como a quantidade de ligações iniciais na inclusão de um novo ator na rede e o valor mínimo de centralidade para utilizar como critério de remoção de um ator da rede (e seu correspondente classificador do conjunto). Nos experimentos com DWM foram utilizados os seguintes parâmetros: β=0,5; θ=0,01; p=1.

No modelo de rede livre de escalas de Barabási (seção 2.5.3), na inclusão de um novo ator na rede, é criada uma quantidade inicial de relacionamentos deste ator com outros atores da rede, representado por *m*. Na Figura 31 são apresentados os desempenhos da metodologia CCS utilizando valores diferentes para este parâmetro (os testes foram feitos utilizando o 3-NN como algoritmo base de aprendizagem e representam a média de 50 iterações) e na Figura 32 são apresentadas as quantidades de classificadores no conjunto. Os valores testados foram: *m=1*, *m=2*, *m=3* e *m=média dos ligações dos atores na rede*. Os resultados mostram que neste domínio o melhor valor é *m=2*, uma vez que é obtido um melhor desempenho com menor quantidade de classificadores. As figuras representam os experimentos que foram mais conclusivos para definir o valor do parâmetro. Os demais experimentos para definição de *m* estão no Anexo III.

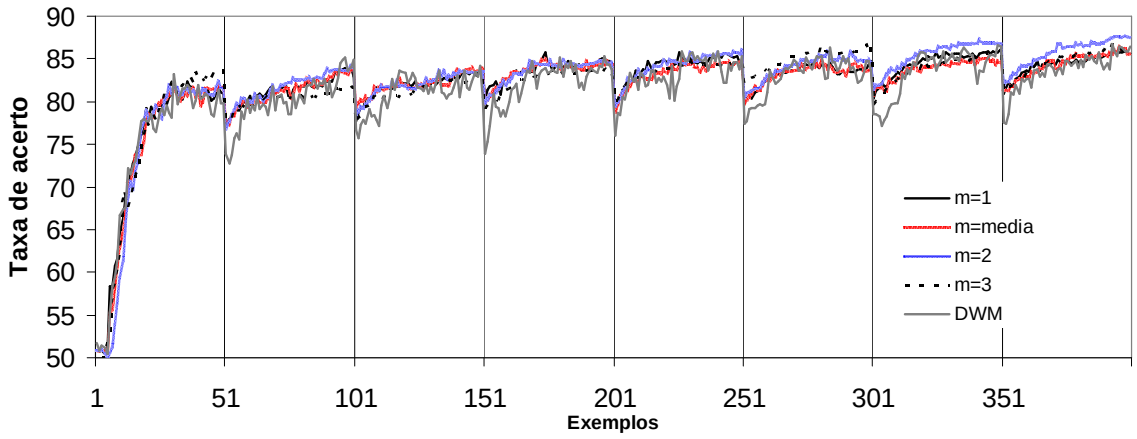


Figura 31 – Taxa de acerto - Mudança moderada (3-NN)

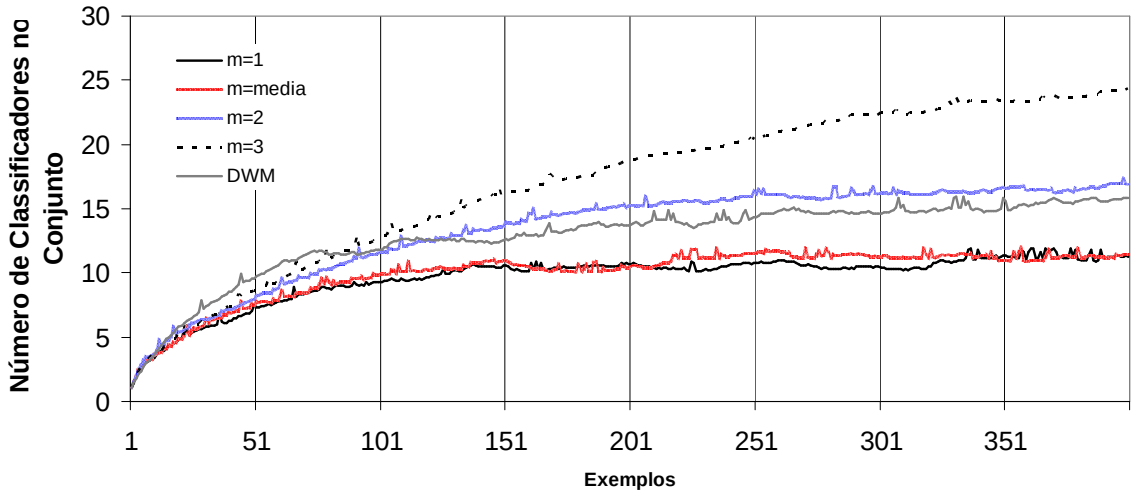


Figura 32 – Quantidade de classificadores - Mudança moderada (3-NN)

As gráficos das figuras 33 e 34 apresentam uma comparação entre os resultados obtidos com valores diferentes de centralidade para a remoção de um classificador (utilizando o 3-NN como algoritmo base de aprendizagem). Observa-se que com um *threshold* menor (*25%* da maior centralidade), a quantidade de classificadores no conjunto é maior. Utilizando um *threshold* maior (*50%* da maior centralidade), a quantidade de classificadores diminui consideravelmente. O desempenho, contudo, não se altera tanto, sendo que em determinados trechos cada estratégia de remoção implica em melhor desempenho. O valor de centralidade

escolhido para remoção de classificadores neste trabalho é *30%* da maior centralidade dentro da rede. Quando comparado com o valor de *40%* da maior centralidade, o desempenho é quase o mesmo, porém, a quantidade de classificadores diminui consideravelmente. No caso do valor de *25%*, o desempenho também é praticamente o mesmo, porém, a quantidade de classificadores aumenta muito. Os resultados para este experimento com os tipos de mudança abrupta e moderada encontra-se no Anexo IV.

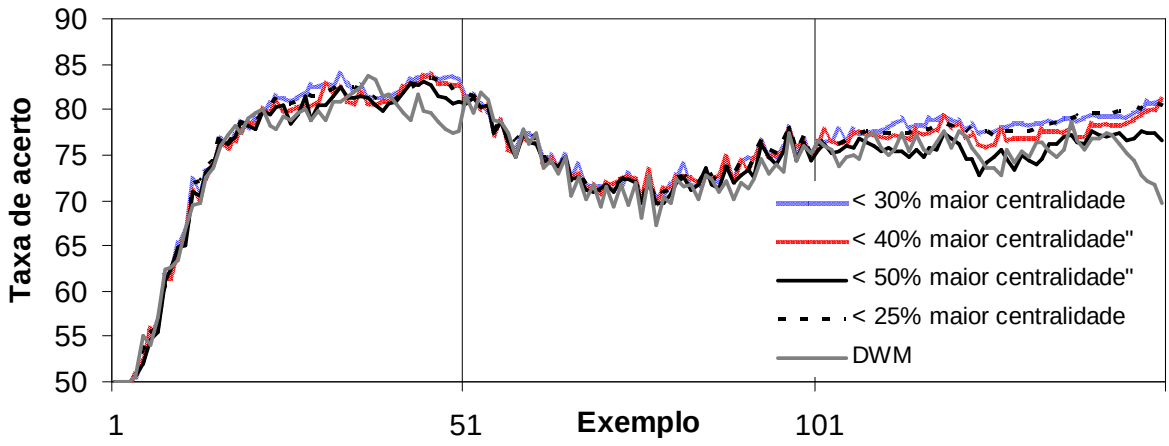


Figura 33 – Taxa de acerto - Mudança Gradual (3-NN)

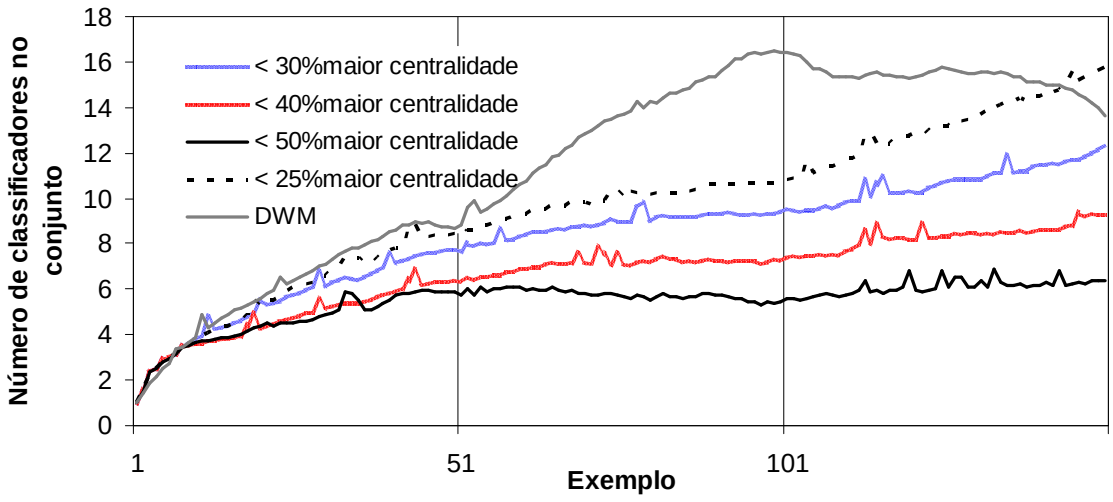


Figura 34 – N° de classificadores - Mudança Gradual (3-NN)

4.2 Comparação CCS X DWM

Nesta seção serão detalhados e comparados os resultados obtidos com a metodologia proposta e a metodologia DWM. Estes resultados representam a média de 50 experimentos. A geração do conjunto é homogênea (Moreira, et al.,2009), uma vez que o algoritmo usado para indução dos classificadores é o mesmo. A comparação entre as metodologias será feita utilizando um algoritmo de aprendizagem simbólica (J48) e um algoritmo de aprendizagem baseada em exemplo (3-NN).

Os parâmetros utilizados no algoritmo CCS foram definidos através dos experimentos apresentados anteriormente. Para a quantidade de ligações iniciais foi utilizado *m=2*. Para o critério de remoção de classificadores, a centralidade utilizada foi *30%* da maior centralidade da rede (*θ=0,3*).

No caso do DWM, os parâmetros utilizados foram: β=0,5; θ=0,01; *p*=1.

Os dados de treinamento e teste foram distribuídos da seguinte forma:

- Mudança abrupta: foram definidos 5 conceitos (Tabela 6) e, para cada conceito, 50 exemplos de treinamento do conceito (positivos e negativos) foram criados, de forma que a cada 50 exemplos o conceito muda. Para cada conceito, um arquivo para testes com 50 exemplos do conceito (positivos e negativos) foi criado.

- Mudança moderada: foram definidos 8 conceitos (Tabela 7) e, da mesma forma que na mudança abrupta, foram gerados arquivos de treinamento e teste com 50 exemplos (positivos e negativos) para cada conceito.

- Mudança gradual: os arquivos de treino foram compostos com 50 exemplos (positivos e negativos) classificados de acordo com um conceito *A*. Até o exemplo 50, α = 1. Após o exemplo de número 50, α começa a diminuir, de forma que o conceito A começa a ser substituído por um conceito *B*. No exemplo 100, o conceito *B* torna-se estável (α = 0) e assim permanece até o final do experimento. Na zona de mudança, a distribuição dos exemplos por conceitos é feita de acordo com a probabilidade *p(A)* e *p(B) = 1 – p(A)*, sendo que *p(A)* decresce enquanto *p(B)* aumenta. Os arquivos de teste, para a parte estável dos dados, seguem os mesmos moldes dos outros tipos de mudança: 50 exemplos de teste para cada conceito. Porém, como visto anteriormente, na zona de *drift* a distribuição dos exemplos nos arquivos de teste deve refletir a distribuição da qual o conceito do exemplo corrente de treinamento foi escolhido. Isto significa que, para cada exemplo de treinamento na zona de *drift*, deve haver

um arquivo de testes com 50 exemplos, distribuídos de acordo com *p(A)* e *p(B)*. Os conceitos A e B estão apresentados na Tabela 8).

A seguir, serão apresentados os resultados obtidos. Os pontos de mudança de conceito são representados por uma linha vertical nos gráficos.

4.2.1 Mudança Abrupta

A Figura 35 representa os resultados dos experimentos da aplicação das metodologias CCS e DWM em dados com mudança abrupta de conceito utilizando o algoritmo 3-NN como base de aprendizagem. Apesar de um desempenho similar das duas metodologias, o CCS apresenta uma recuperação mais rápida no desempenho após a mudança de conceito.

Observa-se também que na metodologia CCS a taxa de acerto é mais constante. Isto pode ser explicado por uma menor variação nos classificadores do conjunto: entre os exemplos do mesmo conceito, a base de classificadores do conjunto se mantém a mesma.

Quando o algoritmo base de aprendizagem é o J48 (Figura 36) o comportamento é o mesmo.

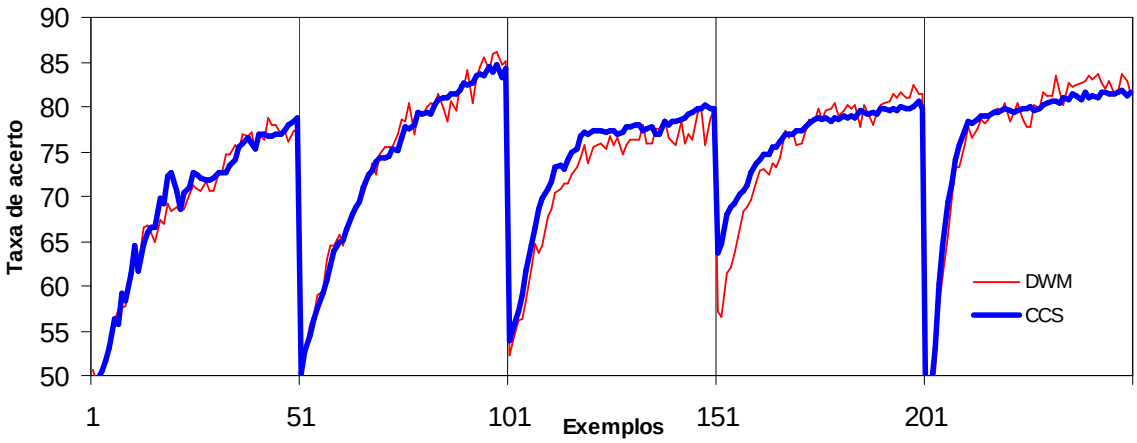


Figura 35 – Taxa de acerto CCS X DWM – Mudança abrupta (3-NN)

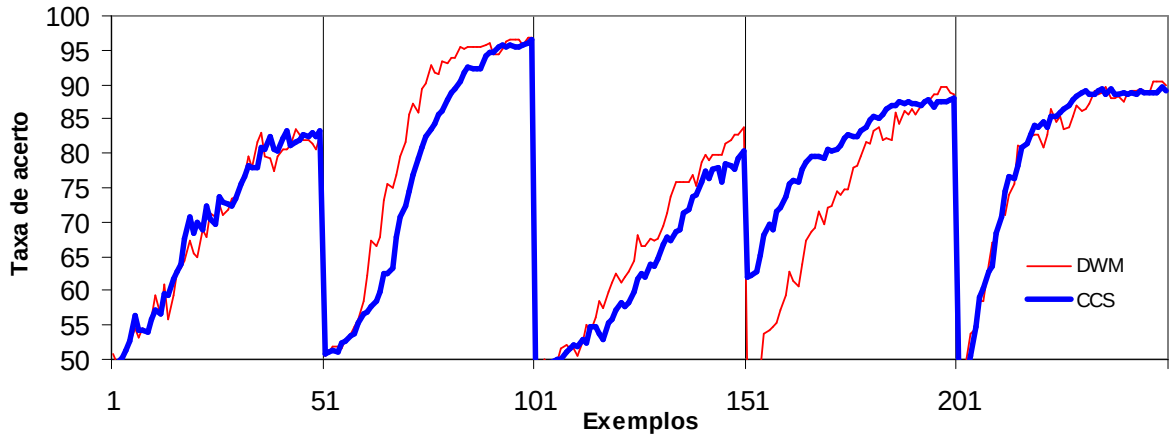


Figura 36 – Taxa de acerto CCS X DWM – Mudança abrupta (J48)

Nos gráficos das figuras 37 e 38 estão representadas as quantidades de classificadores no conjunto utilizando os algoritmos 3-NN e J48, respectivamente. Em ambos os casos, a quantidade de classificadores é menor no caso do CCS. Nestes gráficos são representados os desvios padrão para a média de classificadores entre os experimentos. Os valores mais baixos do desvio do CCS mostram que a metodologia apresenta uma variação menor entre os resultados dos experimentos, o que indica que é mais estável e, portanto, mais confiável que o DWM. No Anexo V são apresentados os gráficos referentes ao total de classificadores criados durante a construção do conjunto.

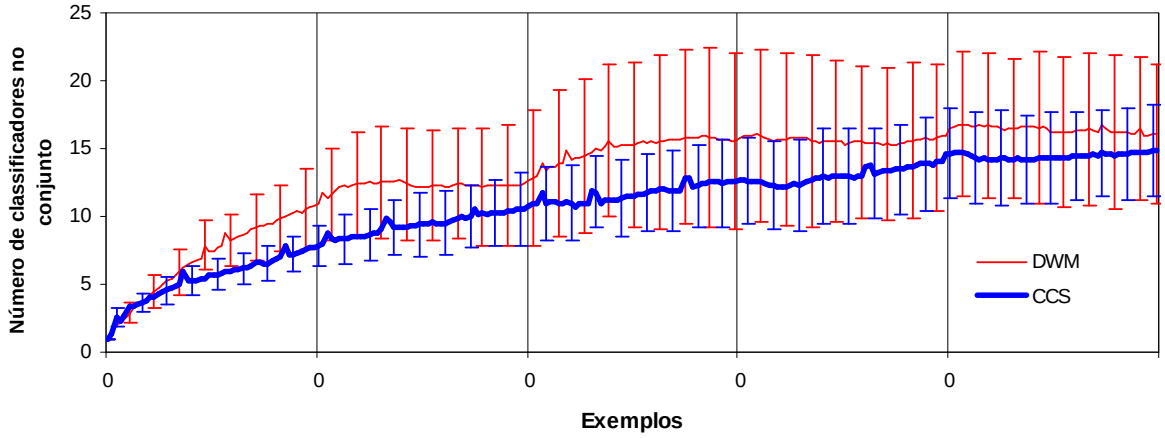


Figura 37 - N° de classificadores CCS X DWM - Mudança abrupta (3-NN)

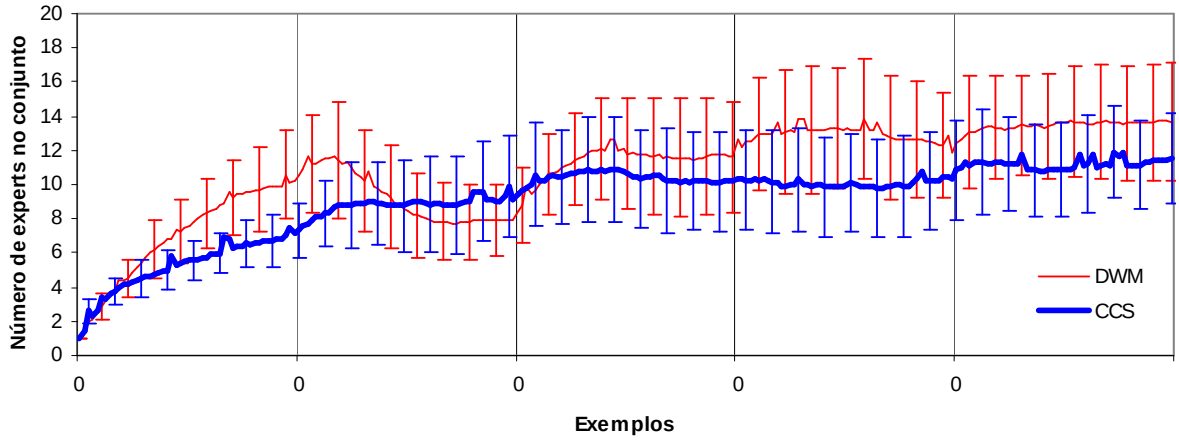


Figura 38 - N° de classificadores CCS X DWM - Mudança abrupta (J48)

As figuras 39 e 40 mostram que a idade média dos classificadores do conjunto é maior na metodologia CCS. Isto significa que a metodologia é mais conservadora que o DWM, uma vez que favorece os classificadores mais antigos. No caso do DWM, o classificador nasce com o peso máximo (que diminui à medida que o classificador erra na predição), o que favorece os novos classificadores a permanecerem no conjunto. No caso do CCS, a centralidade aumenta à medida que o classificador acerta na predição, o que faz com que os classificadores mais antigos permaneçam no conjunto.

A vantagem de uma metodologia mais conservadora é que um classificador com um histórico de bom desempenho não é descartado por um baixo desempenho durante um curto período de treinamento. Por outro lado, novos classificadores demoram para atingir um nível elevado de "maturidade" (número de conexões), o que pode ocasionar sua remoção prematura do classificador do conjunto.

Outro fato que pode ser observado nestes gráficos é a queda da idade média dos classificadores após a mudança de conceito (destacado por círculos). Isto é explicado pelo fato de que os classificadores antigos erram mais na predição, uma vez que o conceito está mudando, o que causa a remoção destes classificadores. À medida que estes classificadores são substituídos por novos classificadores, a idade média do conjunto diminui.

Os gráficos das figuras 41 e 42 apresentam a média de idade dos classificadores removidos e comprovam a idéia de que no CCS prevalecem os classificadores mais antigos.

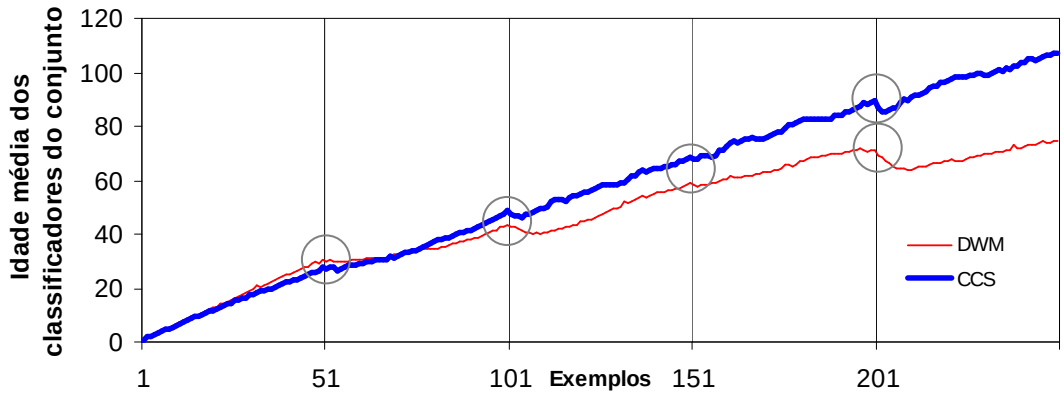


Figura 39 – Idade média dos classificadores CCS X DWM - Mudança abrupta (3-NN)

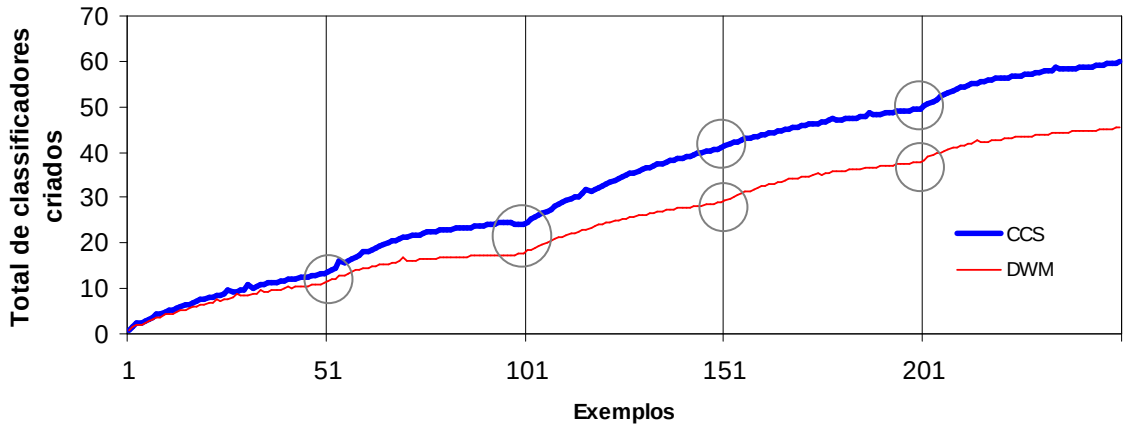


Figura 40 – Idade média dos classificadores CCS X DWM - Mudança abrupta (J48)

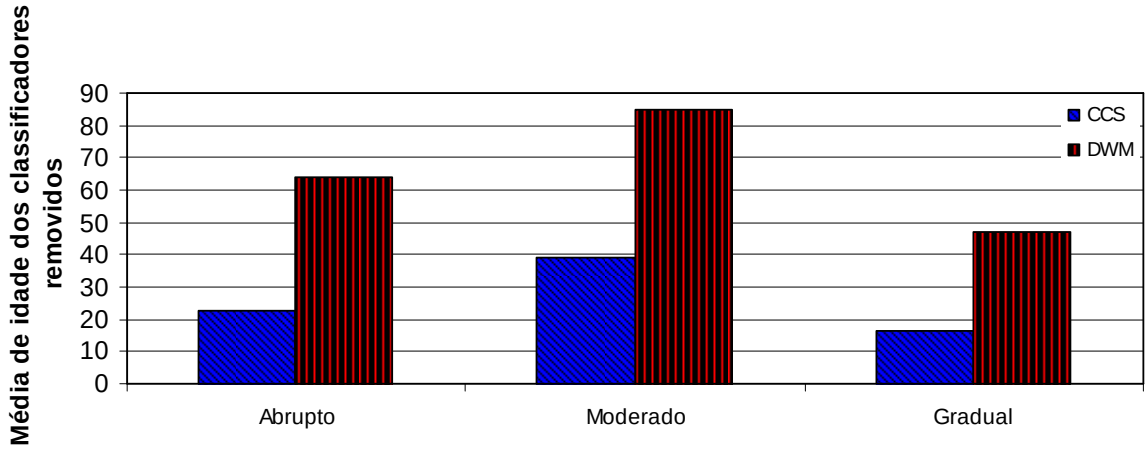


Figura 41 – Idade média dos classificadores removidos (3-NN)

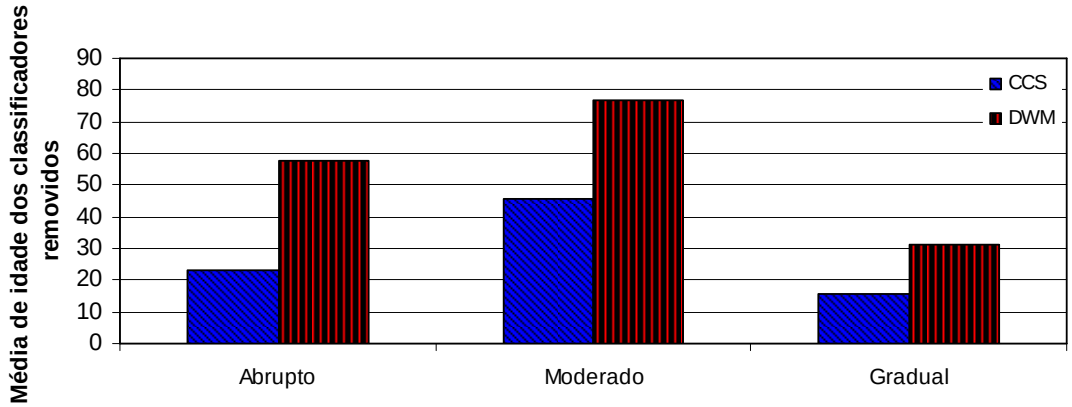
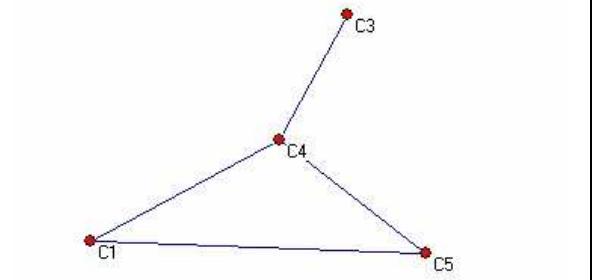
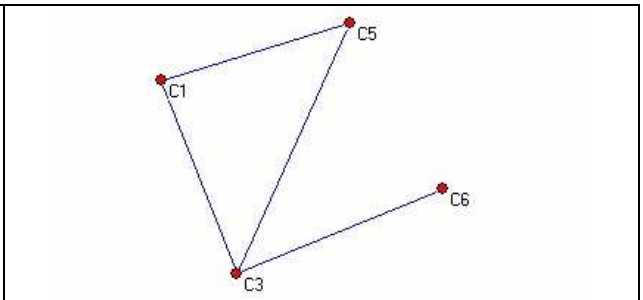


Figura 42 – Idade média dos classificadores removidos (J48)

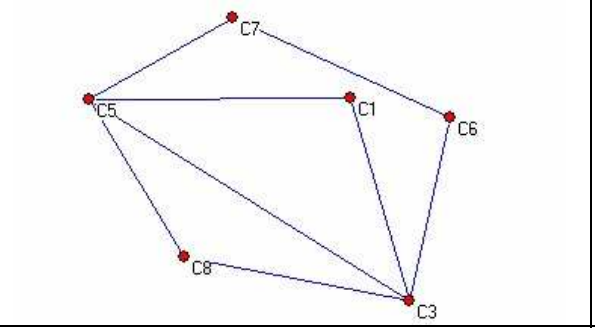
A sequência de grafos da Figura 43 mostra as redes de relacionamentos criadas durante a construção do conjunto. Percebe-se que a base do conjunto é formada pelos mesmos classificadores até que ocorra a mudança de conceito. Entre o grafo do exemplo 50 e o grafo do exemplo 100, todo o conjunto de classificadores é substituído. Do exemplo 100 ao 150, somente o classificador C_{12} permanece no conjunto.



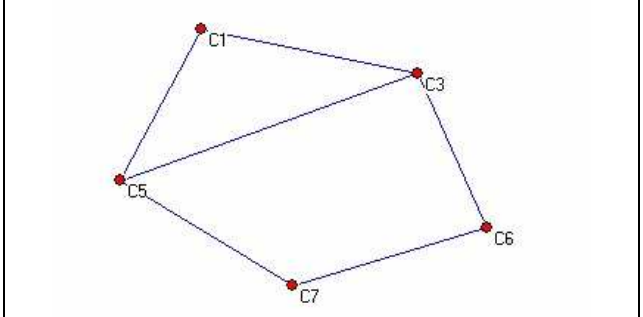
Exemplo 10



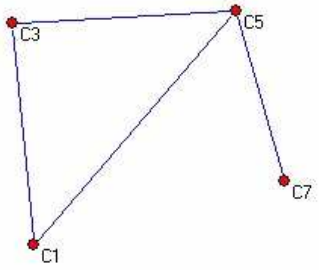
Exemplo 20



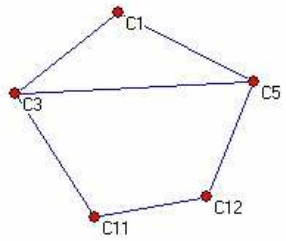
Exemplo 30



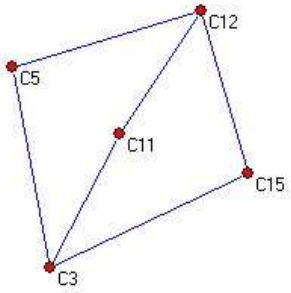
Exemplo 40



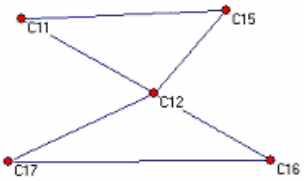
Exemplo 50



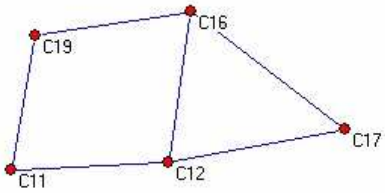
Exemplo 60



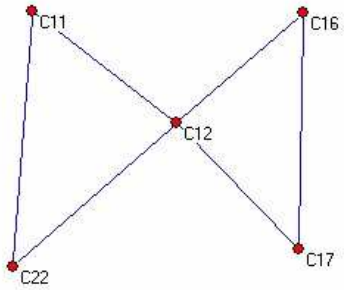
Exemplo 70



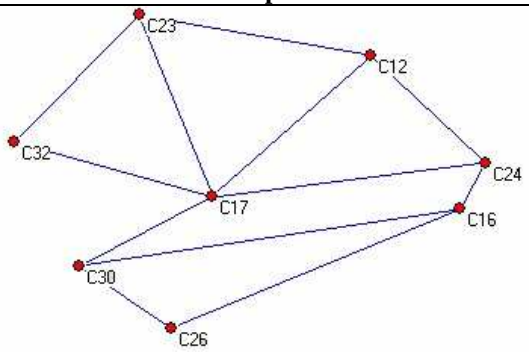
Exemplo 80



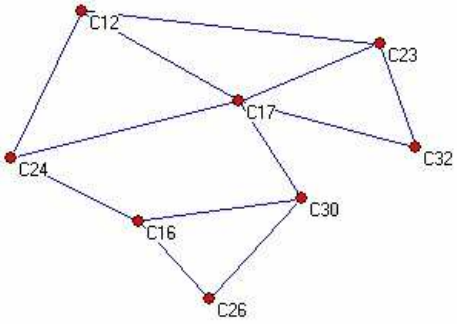
Exemplo 90



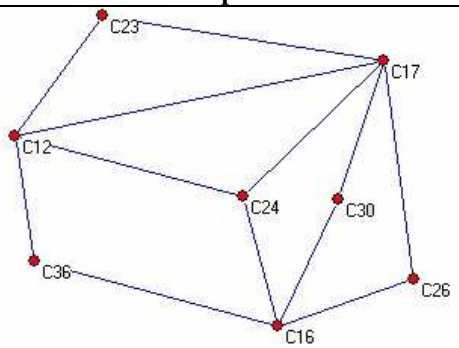
Exemplo 100



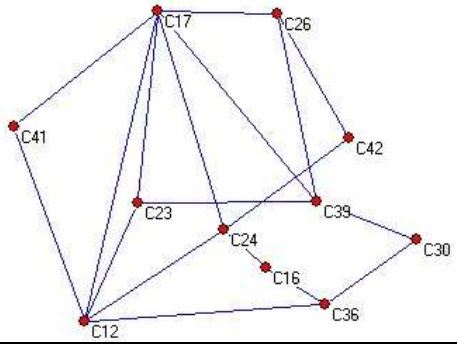
Exemplo 140



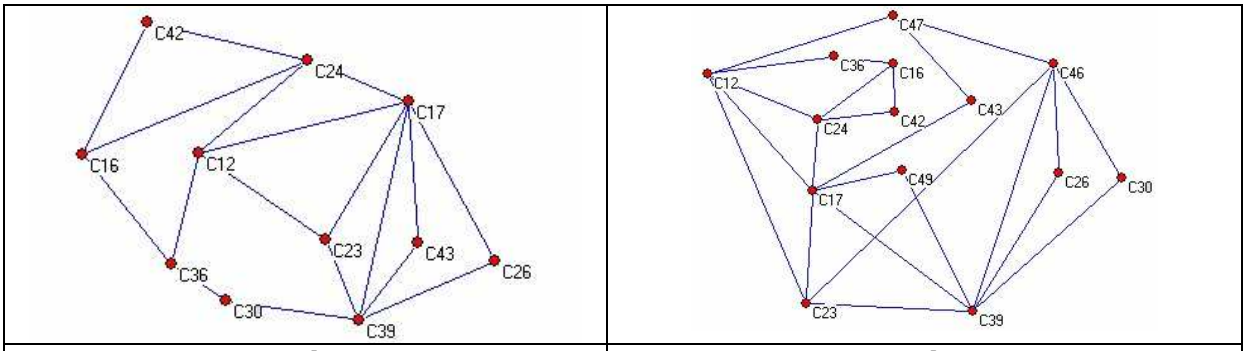
Exemplo 150



Exemplo 160



Exemplo 185



Exemplo 200	Exemplo 215

Figura 43 – Evolução da rede de relacionamentos - Mudança abrupta (3-NN)

É possível observar que, apesar de utilizar a anexação preferencial, a rede criada não apresenta uma característica de uma rede livre de escalas (alguns classificadores com muitas ligações, os *hubs*). Isto pode ser explicado pelo fato de que, além da anexação preferencial, a atração de novos relacionamentos também depende do desempenho do classificador. Ou seja, os atores somente atraem mais relacionamentos quando predizem corretamente os exemplos de treinamento. Outro fator que contribui para isto é a remoção dos classificadores à medida que estes atingem uma centralidade mínima. Remover um classificador implica em remover os seus relacionamentos, o que diminui a centralidade de outros classificadores.

4.2.2 Mudança moderada

No caso da mudança moderada (situação em que cada conceito mantém similaridade com o conceito anterior) não ocorre uma queda brusca no desempenho, como no caso da mudança abrupta. Pode-se observar nos gráficos que, ao contrário da mudança abrupta, o desempenho é recuperado mais rapidamente. No caso do 3-NN, o desempenho do CCS quando comparado ao DWM melhora com a quantidade de conceitos (Figura 44). Com o J48 as duas metodologias conseguem praticamente o mesmo desempenho (Figura 45).

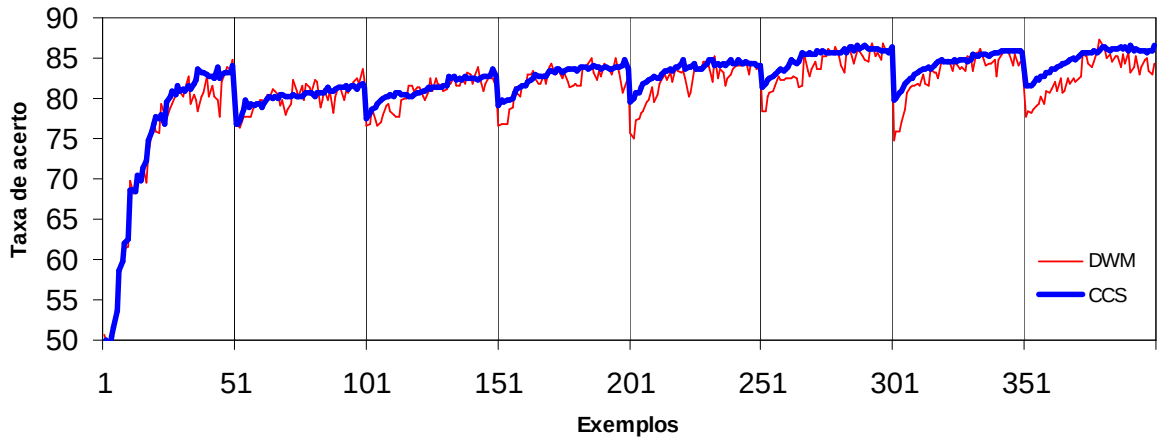


Figura 44 – Taxa de acerto CCS X DWM - Mudança Moderada (3-NN)

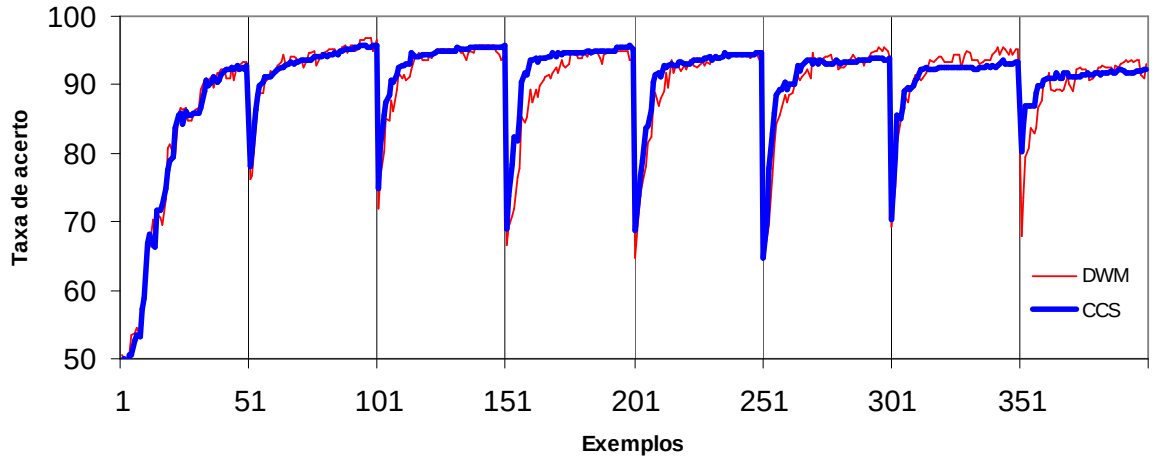


Figura 45 – Taxa de acerto CCS X DWM - Mudança Moderada (J48)

A Figura 47 mostra que a quantidade de classificadores no conjunto para o CCS é praticamente a mesma do DWM quando utilizado o algoritmo J48. No entanto, com o algoritmo 3-NN (Figura 46), é necessária uma quantidade maior de classificadores para conseguir um desempenho melhor que o DWM. As duas figuras mostram que, da mesma forma que na mudança abrupta, o CCS é mais estável que o DWM, uma vez que os resultados obtidos nos experimentos são mais precisos.

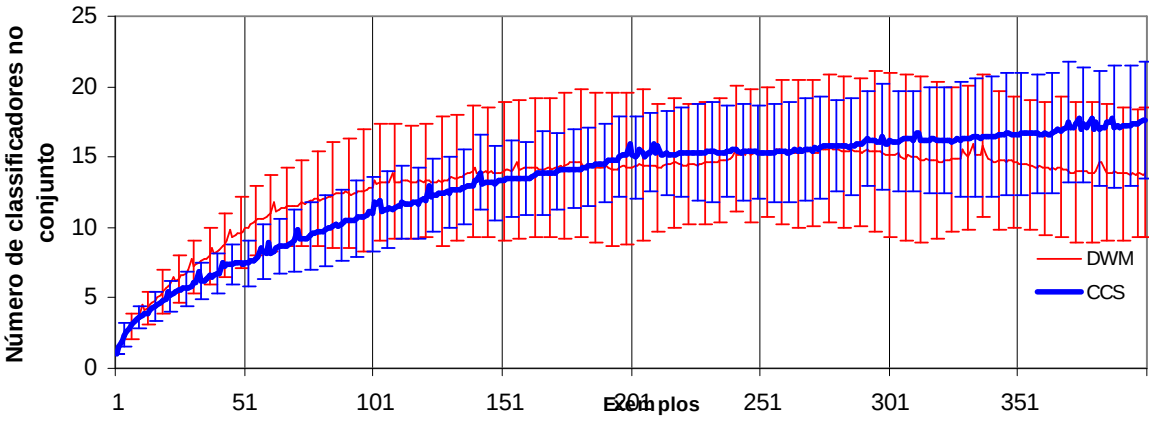


Figura 46 – N° de classificadores CCS X DWM - Mudança moderada (3-NN)

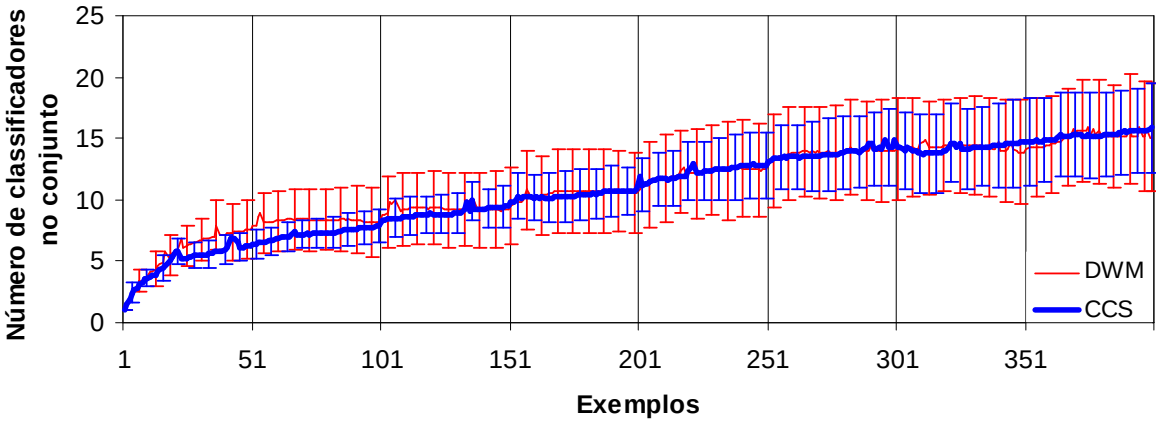


Figura 47 - N° de classificadores CCS X DWM – Mudança moderada (J48)

A média de idade dos classificadores para o CCS, no caso do J48 é praticamente a mesma do DWM (Figura 49). Isto significa que nesta situação os classificadores novos são mais favorecidos tanto no CCS como no DWM. No caso do 3-NN, o CCS continua sendo mais conservador que o DWM, favorecendo os mais antigos (Figura 48).

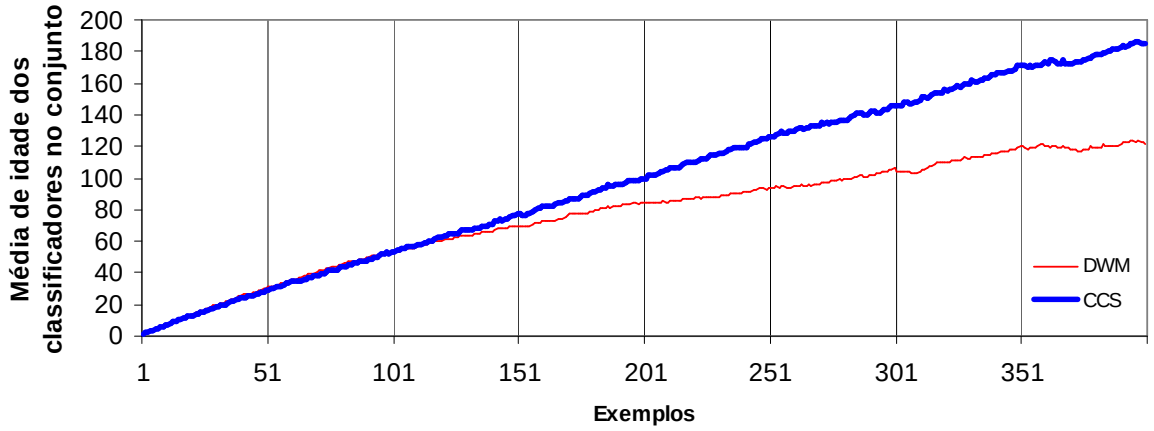


Figura 48 – Idade média dos classificadores CCS X DWM - Mudança moderada (3-NN)

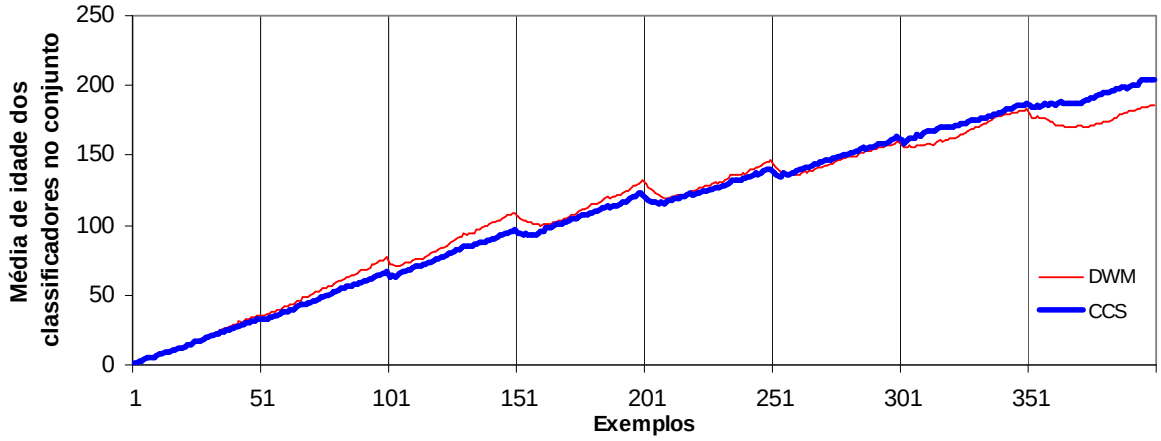


Figura 49 – Idade média dos classificadores CCS X DWM - Mudança moderada (J48)

4.2.3 Mudança gradual

Da mesma forma que na mudança moderada, não há uma queda abrupta no desempenho dos sistemas CCS e DWM na mudança gradual. Neste tipo de mudança, o desempenho do CCS é superior ao do DWM, como pode ser constatado nos gráficos (figuras 50 e 51).

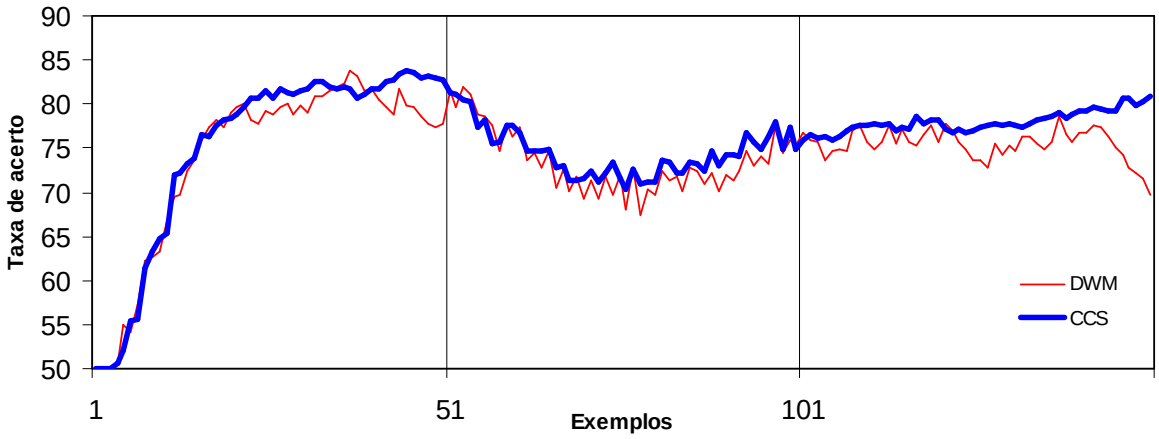


Figura 50 – Taxa de acerto CCS X DWM - Mudança gradual (3-NN)

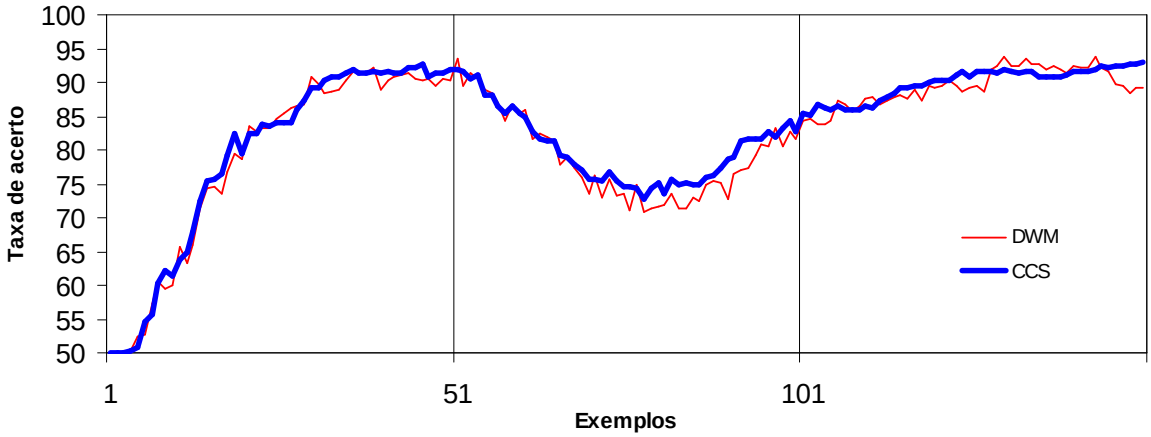


Figura 51 - Taxa de acerto CCS X DWM - Mudança gradual (J48)

As figuras 52 e 53 mostram que o número de classificadores no conjunto é menor no caso do CCS (embora no caso do J48 a quantidade de classificadores é menor para o DWM no final do segundo conceito) e este número se mantém constante. Observa-se que no caso do DWM, a quantidade de classificadores diminui no final do segundo conceito, mas isto acarreta uma diminuição na taxa de acertos. Da mesma forma que nas mudanças abrupta e moderada, a variação na quantidade de classificadores entre os experimentos para o CCS foi menor que no caso do DWM.

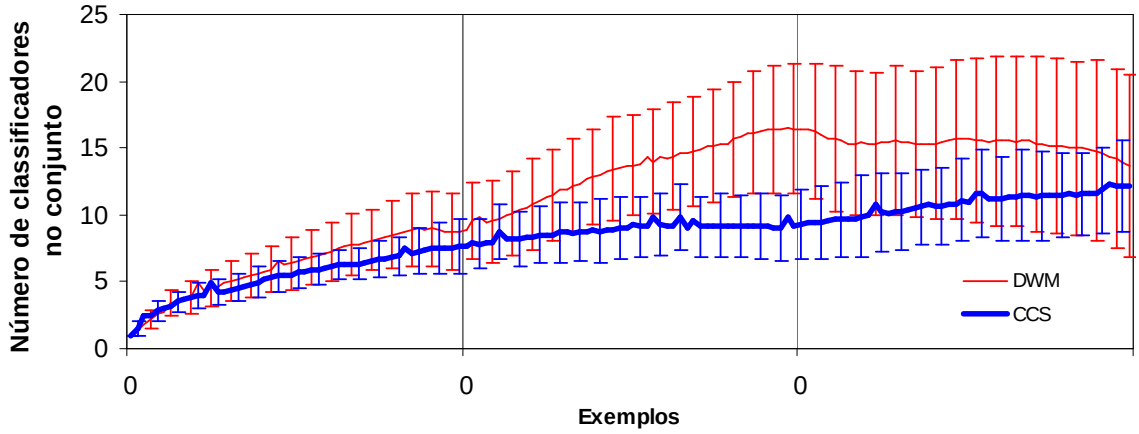


Figura 52 – N° de classificadores CCS X DWM - Mudança gradual (3-NN)

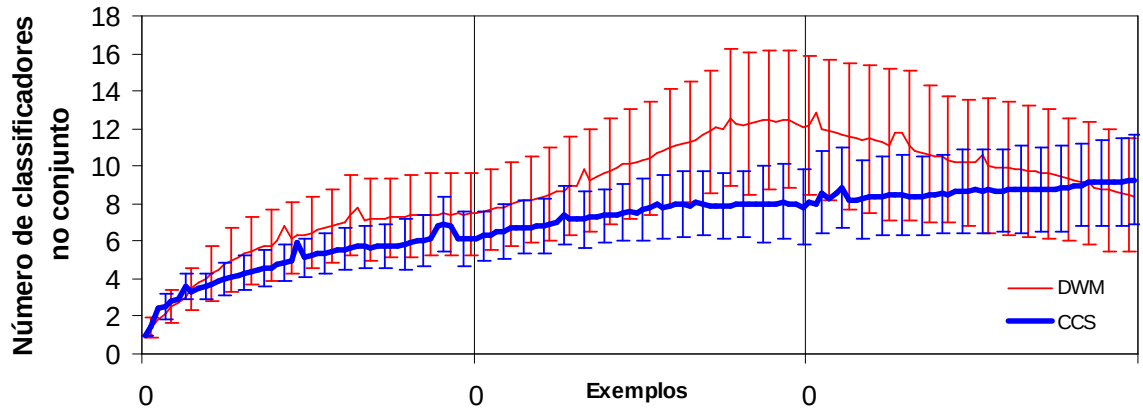


Figura 53 – N° de classificadores CCS X DWM - Mudança gradual (J48)

Ao contrário das mudanças abrupta e moderada, na mudança gradual a média de idade dos classificadores no conjunto no CCS e no DWM é muito próxima (figuras 54 e 55). No caso do J48, quando o segundo conceito se estabiliza, o CCS torna-se inclusive menos conservador que o DWM e mantém os classificadores mais novos. Observar que a queda de desempenho do DWM foi resultado da queda na quantidade de classificadores. No caso do CCS, a quantidade de classificadores se manteve praticamente constante enquanto o desempenho melhorou.

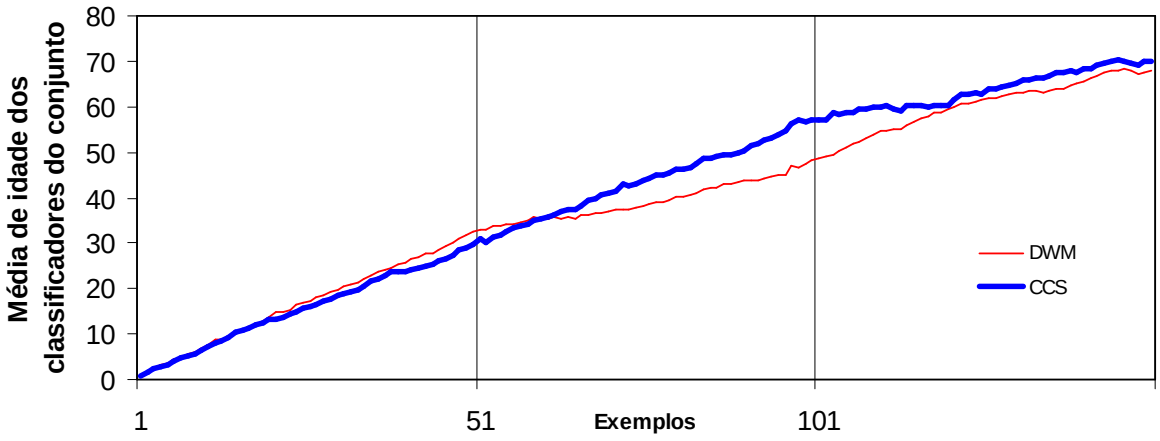


Figura 54 – Idade média dos classificadores CCSX DWM – Mudança Gradual (3-NN)

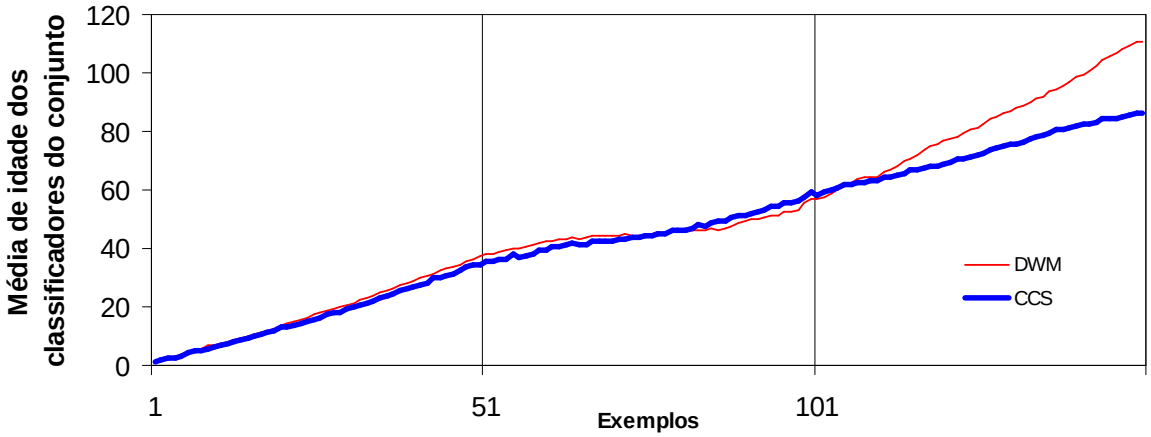


Figura 55 – Idade média dos classificadores CCS X DWM – Mudança gradual (J48)

4.3 Considerações finais

Nesta seção foram apresentados os resultados obtidos com a aplicação das metodologias CCS e DWM em uma base de dados artificial, que simula o *concept drift*. De modo geral, os resultados demonstraram a eficiência de uma abordagem social na construção e integração de um conjunto de classificadores. Na comparação com o DWM, o CCS apresentou um desempenho equivalente ou superior com uma quantidade menor de classificadores.

5 CONCLUSÃO

Esta pesquisa teve como objetivo verificar a aplicação dos conceitos da análise de redes sociais na integração dos classificadores de um conjunto, especificamente num domínio com *concept drift*. Partindo da hipótese que um conjunto de classificadores pode ser modelado como uma rede de relacionamentos, onde os classificadores são os atores e os relacionamentos entre eles representam a predição idêntica para um exemplo, foi desenvolvida uma metodologia denominada Conjunto de Classificadores Sociais (CCS).

Para esta metodologia o modelo de rede livre de escalas foi o modelo escolhido para criação da rede de relacionamentos devido às características semelhantes deste modelo com conjunto de classificadores: no conjunto, os classificadores possuem reputações diferentes, as ligações entre os membros são aleatórias e a quantidade de indivíduos pode variar em função das escolhas coletivas realizadas.

A principal característica deste modelo é que a probabilidade de que um ator atraia novos relacionamentos é proporcional à sua centralidade (anexação preferencial). A construção da rede durante a etapa de treinamento, além da inclusão de novo ator na rede envolve os seguintes procedimentos:

1. de acordo com alguns critérios baseados nos desempenhos individuais dos classificadores, alguns atores alteram a sua posição na rede, substituindo ligações com alguns atores por ligações com outros atores. O objetivo desta adaptação é obter um melhor desempenho do conjunto.

2. quando algum ator atinge uma posição dentro da rede que o torna irrelevante nas decisões coletivas, este ator é removido. Esta posição é verificada utilizando a ARS.

Uma vez construída uma rede de relacionamentos, a utilização de conceitos da ARS permite definir as reputações dos classificadores dentro desta rede de relacionamentos. Estas reputações vão indicar o peso do classificador nas decisões do conjunto.

Portanto, a ARS é utilizada na fase de construção do conjunto, definindo os classificadores que podem ser removidos ao perder sua relevância, e na fase de integração dos classificadores, para a definição das decisões coletivas.

Para avaliar a metodologia proposta foram realizados diversos experimentos em ambiente com *concept drift*. Os experimentos foram conduzidos utilizando uma base de dados artificial, representando um domínio de negociação bilateral. Esta base foi criada simulando os três tipos de mudança: abrupta, gradual e moderada. Além disto, a metodologia foi avaliada

utilizando os algoritmos 3-NN e J48 como algoritmo base de aprendizagem, representantes de algoritmos baseados em exemplos e simbólicos, respectivamente.

Nos experimentos, a metodologia CCS foi comparada com a metodologia DWM e o seu desempenho foi similar, e até melhor em muitas situações, com menor quantidade de classificadores. A metodologia também apresentou um menor desvio padrão entre os experimentos, o que mostra uma maior estabilidade.

De modo geral, os resultados obtidos foram satisfatórios, o que permite concluir que através de uma rede de relacionamentos construída a partir dos classificadores de um conjunto é possível utilizar os conceitos da análise de redes sociais na construção do conjunto e na integração dos classificadores para decisão coletiva.

No entanto, cabe ressaltar que a evolução da rede não ocorreu da forma esperada: no modelo utilizado (livre de escalas), a rede final seria composta por alguns atores (classificadores) com muitos relacionamentos e que seriam os mais relevantes nas decisões coletivas. Entretanto, além da anexação preferencial, na metodologia CCS a atualização da rede de relacionamentos depende também do desempenho individual dos classificadores. Outro fator que influencia no modelo final da rede é o processo de remoção de classificadores: ao remover um classificador, seus relacionamentos são também removidos, o que acaba afetando a centralidade de outros classificadores na rede.

A expectativa inicial era de eliminação de parâmetros para configuração da metodologia, porém, foram necessários parâmetros para indicar a centralidade mínima para remoção de classificadores e também a quantidade inicial de ligações no momento de inclusão de um ator na rede.

Neste trabalho a medida de reputação utilizada foi a centralidade de grau. Em estudos futuros deve ser analisado o comportamento da metodologia utilizando as demais medidas de centralidade da ARS como centralidade de intermediação e informação.

A ARS também fornece diferentes formas de exploração da rede como métricas de grupos, cliques e outros conceitos não explorados neste trabalho que podem ser utilizados futuramente para exploração da metodologia apresentada. Além disto, novas estratégias de adaptação da rede podem ser utilizadas, assim como diferentes heurísticas para criação e remoção de classificadores.

Os experimentos conduzidos neste trabalho indicam que a metodologia proposta pode ter um desempenho superior ao DWM. Outras avaliações são necessárias para confirmar este desempenho. Uma delas seria utilizando bases de dados de domínios diferentes e bases de dados reais.

REFERÊNCIAS BIBLIOGRÁFICAS

Aha, David W.; et al.. Instance-Based Learning Algorithms. *Machine Learning*, n. 6, v. 1, p. 37-66, 1991.

Albert, Réka; Barabási, Albert-László. Topology of Evolving Networks: Local Events and Universality, *Physical Review Letters*, n. 24, v. 85, 2000.

Albert, Réka; Barabási, Albert-László. Statistical mechanis of complex networks, *Reviews of modern physics*, v. 74, Janeiro 2002.

Araujo, Ricardo M.; Lamb, Luis C.. Memetic Networks: Analyzing the Effects of Network Properties in Multi-Agent Performance. *Proceedings of the Twenty-Third AAAI Conference of Artificial Inteligence*, 2008.

Bauer, Eric; Kohavi, Ron. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, e Variants. *Kluwer Academic Publishers*, Boston, vol. 36, p. 105-139, 1999.

Borgatti, Stephen P.; Foster, Pacey C.. The Network Paradigm in Organizational Research: A Review e Typology. *Journal of Management*, v. 29, n.6, p. 991-1013, 2003.

Breiman, Leo. Bagging Predictors. *Machine Learning*, v. 24, p. 123-140, 1996.

Carling, Alison. *Introducing Neural Networks.* Wilmslow, England: Sigma Press. 1992.

Cristianini, Nelo; Shawe-Taylor, John. *Support Vector Machines*. Cambridge University Press, 2000.

Delany, Sarah Jane; et al.. A case-based technique for tracking concept drift in spam filtering. *Science Direct, Knowledge-Based Systems*, v. 18 p. 187-195, 2005.

Dimov, Rossen. Weka: Practical machine learning tools and techniques with Java implementations,. *AI Tools Seminar*,University of Saarland, WS 06/07, 2007.

Dorogovtsev, S.N.; et al.. Structure of Growing Networks with Preferential Linking. *Physical Review Letters*, v. 85, n. 21, 2000.

Dorogovtsev, S.N.; Mendes, J.F.F.. Evolution of reference networks with aging. *Physical Review Letters,* v. 52, n. 33, p. 1842-1845, 2000.

Dorogovtsev, S.N.; et al.. Growing network with heritable connectivity of nodes. Preprint cond-mat/0011077, 2000.

Erdös, Paulo; Rényi, Alfred. 1958. Disponível em: <http://www.renyi.hu/~p_erdos/ 1959-11.pdf>. Acessado em: 01/05/2010.

Enembreck, Fabrício; et al.. Learning Drifting Negotiations. *Applied Artificial Intelligence*, v. 21, n. 9, p. 861-881, Outubro 2007.

Enembreck, Fabrício; et al.. Learning Negotiation Policies Using Ensemble-based Drift Detection Techniques. *International Journal on Artificial Intelligence Tools*, v.18, n. 2 p. 173-196, 2009.

Freund, Yoav; Schapire, Robert E.. A decision-theoretic generalization of on-line learning and an aplication to boosting. *Computacional Learning Theory: EuroCOLT '95*, p.23-3, 1995.

Freund, Yoav; Schapire, Robert E.. Experiments with a New Boosting Algorithm. *Thirteenth International Conference on Machine Learning*, pp. 148-156, 1996.

Freund, Yoav; Schapire, Robert E.. A Short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence*, v. 14, n. 5, p. 771-780, Setembro 1999.

Gama, João; et al.. Learning with Drif Detection. *SBIA Brazilian Symposium on Artificial Intelligence*, v. 3171, p. 286-295, Outubro 2004.

Gaston, Matthew E.; desJardins, Marie. Team Formation in Complex Networks. *Proceedings of the North American Association for Computational Science and Organizational Science (NAACSOS) Conference*, Junho 2003.

Gaston, Matthew E.; et al.. Adapting Network Structure for Efficient Team Formation. *Proceedings of the AAMAS'04, Workshop on Learning and Evolution in Agent Based System*, Julho 2004.

Gaston, Matthew E.; desJardins, Marie. Social Networks e Multi-agent Organizational Performance. *FLAIRS'05*, Maio 2005.

Gaston, Matthew E.; desJardins, Marie. Agent-organized networks for dynamic team formation. *AAMAS'05*, Utrecht, Netherles, Julho 2005.

Granovetter, Mark. The Strength of Weak Ties: A Network Theory Revisited. *Sociological Theory*, v. 1, p. 201-233.

Haykin, Simon. Neural Networks – A Comprehensive Foundation. *New York: IEEE Press*, 1994.

Indurkhyay , Nitin; Weissz, Sholom M.. Estimating Performance Gains for Voted Decision Trees. *Intelligent Data Analysis (IDA)*, v.2, n.1-4, p.303-310, 1998.

Klinkenberg, Ralf. Learning Drifting Concepts: Example selection vs. example weighting. *Intelligent Data Analysis (IDA)*, v. 8, n. 3, p. 281-300, Agosto 2004.

Knoke, D.; Yang, S.. Social Network Analysis. *Series: Quantitative Applications in the Social Sciences*, Sage Publications Ltda, 2008.

Kohavi, Ron. A Study of Cross Validation and Bootstrap for Accuracy Estimation and Model Selection, *International Joint Conference on Artificial Intelligence*, p. 1137-1143, 1995.

Kohavi, Ron; Quinlan, Ross. Decision Tree Discovery. *Handbook of data mining and knowledge discovery*, p. 267-276, 1999.

Kolter, Jeremy Z.; Maloof, Marcus A.. Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift, *Proc. of 3th International IEEE Conference on Data Mining*, p.123-130, IEEE Press, 2003.

Kolter, Jeremy Z.; Maloof, Marcus A.. Using Additive Expert Ensembles to Cope with Concept Drift. *22th International Conference on Machine Learning*, p. 449-456, ACM Press, 2005.

Kubat, Miroslav; Widmer, Gerhard. Adapting to Drift in Continuous Domains. *Proceedings of the 8th European Conference on Machine Learning*, p. 307-310, 1995.

Link, Hamilton; et al.. The impact of Social Networks on Multi-Agent Recommender Systems. arXiv:cs/0511011v1 [cs.LG] 2/10/2005, 2005.

Littlestone ,Nick; Warmuth, Manfred K.. The weighted majority algorithm. *Information e Computation*, v. 108, n. 2, p. 212-261, 1994.

Marteleto, R. M; Tomaél, M. I.. A metodologia de análise de redes sociais (ARS). In.: Valentim, M.L.P.(Org.). *Métodos qualitativos de pesquisa em ciência da informação*, São Paulo: Polis, 2005.

Merida-Campos, Carlos; Willmott, Steven. The impact of betweenness in small world networks on request for proposal coalition formation problems. *CCIA'07*: Sant Julia de Loria, Eorra, 2007.

Mitchel, Tom M.. *Machine Learning*, McGraw-Hill, 1997.

Moreira, João M.; et al.. Ensemble learning: a study on different variants of the dynamic selection approach. *Proceeding MLDM'09 Proceedings of the 6th International Conference on Machine Learning and Data Mining in Pattern Recognition*, 2009.

Newman, M.E.J.. The mathematics of networks. *Siam Review*, Philadelphia, v. 45, n.2, p.167-256, 2003.

Newman, M.E.J.. The structure e function of complex networks. *Siam Review*, Philadelphia, v.45, n.2, p.167-256, 2003.

Nicoletti, Maria C.. Ampliando os Limites do Aprendizado Indutivo de Máquina através de Abordagens Construtiva e Relacional, Tese apresentada ao Instituto de Física de São Carlos – USP, São Carlos, 1994.

Okamoto, Kazuya; et al.. Ranking of Closeness Centrality for Large-Scale Social Networks. *In Frontiers in Algorithmics*, v. 5059, p. 186-195, 2008.

Page, Lawrence; et al.. The PageRank Citation Ranking: Bringing Order to the Web, *Stanford Digital Library Technologies Project*. Disponível em < http://dbpubs.stanford.edu/pub/1999-66>. Acessado em 29/07/2011.

Pujol, J.M., Sangüesa, R., Delgado, J.. Extracting Reputation in Multi Agent Systems by Means of Social Network Topology. AAMAS '02, Bologna, Italy (2002)

Puuronen, Seppo; et al.. A Dynamic Integration Algorithm with Ensemble of Classifiers. *Proceeding ISMIS'99 Proceedings of the 11th Internation Symposiym on Foundations of Intelligent Systems* Springer-Verlag London, 1999.

Prati, Ronado C.. Novas abordagens em Aprendizado de Máquina para Geração de Regras, Classes Desbalanceadas e Ordenação de Casos, Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC, USP. Maio 2006.

Quinlan, J.R.. Induction of Decision Trees. *Machine Learning*, v.1, n.1, p.81-106, 1986.

Quinlan, J.R.. C4.5 Programs for Machine Learning. *Morgan Kaufmann Publishers*, 1993.

Quinlan, J.R.. Bagging, Boosting, e C4.5. *Proceedings of the Thirteenth National Conference on Artificial Interlligence*, AAAI Press, p. 725-730, 1996.

Ribeiro, Carlos H. C.. A Inteligência Computacional no Estudo de Sociedades e Redes Complexas: Modelos e Abordagens, II Workshop de Dissertações, Pós-Graduação em Ciência da Computação, UFU, 2003.

Ribeiro, Richardson. Análise do impacto da teoria das redes sociais em técnicas de otimização e aprendizagem multiagente baseadas em recompensas. Curitiba : s.n., 2010.

Rosa, Antonio M.. Elementos para uma teoria geral das redes. Disponível em: <http: //www.cecl.com.pt/pdf/amr.pdf>. Acessado em 27/07/2011.

Russel, Stuart; Norvig, Peter. *Inteligência Artificial*, Editora Campus, 2004.

Schapire, Robert E.. The Boosting Aproach to Machine Learning: An Overview, *MSRI Workshop on Nonlinear Estimation and Classification*, USA, 2002.

Schilimmer, Jeffrey C.; Granger, Richard H.. *Beyond Incremental Processing: Tracking concept drift*. Disponível em: <// http://www.aaai.org/Papers/AAAI/1986/AAAI86-084.pdf>. Acessado em: 26/06/2011.

Scott, John. *Social Network Analysis*. Sage Publications Ltda. 2000.

Shinoda, Kosuke; et al.. Emergence of Global Network Property based on Multi-agent Voting Model. *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems(AAMAS '07)*, 2007.

Skurichina, Marina; Duin, Robert P. W.. Bagging, Boosting and the Random Subspace Method for Linear Classifiers. *Pattern Analysis & Applicationst*, v.5, n.2, p.121-135, 2002.

Stanley, Kenneth O.. Learning Concept Drift with a Committee of Decision Trees. *Technical Report*, Department of Computer Sciences, University of Texas at Austin, *AI-03-302*, Setembro 2003.

Stephenson, T.A.. An introduction to Bayesian network theory and usage, *IDIAP Research Report 00-03*. Fevereiro 2000.

Street, W. Nick; Kim, YongSeog. A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge, Discovery and Data mining.* ACM Press, p. 377-382, 2001.

Tsymbal, Alexey. The Problem of concept drift : definitions and related work. *Technical Report TCD-CS-2004-15*, Department of Computer, Trinity College Dublin, Ireland. Abril 2004.

Tsymbal, Alexey; et al.. Dynamic Integration of Classifiers for Tracking Concept Drift in Antibiotic Resisteance Data. Department of Computer Science, Trinity College Dublin, Irele, 2005.

Wasserman, Stanley; Faust, Katherine. *Social Network Analysis – Methods e Applications*, Cambridge University Press. 1994.

Weiss, Gerhard. Multiagent Systems – A modern approach to distributed artificial intelligence, *The MIT Press,* Cambridge, Massachusetts (London, England), 1999

Widmer, Gerhard; Kubat, Miroslav. Effective learning in dynamic environments by explicit context tracking. *European Conference on Machine Learning*, p. 227-243, 1993.

Widmer, Gerhard; Kubat, Miroslav. Learning in the Presence of Concept Drift e Hidden Contexts, *Machine Learning*, v. 23, n. 1, p. 60-101, 1996.

Witten, Ian H.; Frank, Eibe. *Data Mining – Practical Machine Learning Tools and Techniques,* 2 ed. São Francisco: Elsivier, 2005.

Yu-Quan, Zhu; et al.. Dynamic weighting ensemble classifiers based on cross-validation. *Neural Computing & Applications*, v. 20, n. 3, p. 309-317, 2011.

ANEXOS

Anexo I – Exemplo arquivo no formato .arff(*Attribute-Relation File Format*)

```
@relation Oferta Interessante
@attribute cor {preto, azul, ciano, marrom, vermelho, verde, amarelo, magenta}
@attribute preco {muito_baixo, baixo, normal, alto,muito_alto,bastante_alto,enorme,
nao_vendavel}
@attribute pagamento {0,30,60,90,120,150,180,210,240}
@attribute quantidade {muito_baixo,baixo,normal,alto,muito_alto,bastante_alto,enorme,
nao_assegurado}
@attribute entrega {muito_curto,curto,normal,longo,muito_longo}
@attribute interesse {SIM, NAO}
@data
preto,enorme,90,bastante_alto,muito_curto,SIM
preto,enorme,210,muito_baixo,longo,NAO
azul,normal,60,baixo,muito_longo,NAO
preto,muito_baixo,30,muito_baixo,curto,NAO
magenta,alto,120,muito_alto,muito_curto,SIM
magenta,alto,180,muito_baixo,muito_curto,SIM
azul,alto,0,baixo,curto,NAO
verde,baixo,120,bastante_alto,muito_curto,NAO
magenta,alto,120,baixo,muito_curto,SIM
preto,bastante_alto,0,baixo,normal,NAO
ciano,bastante_alto,30,bastante_alto,muito_curto,NAO
magenta,alto,90,nao_assegurado,muito_curto,SIM
ciano,nao_vendavel,180,alto,curto,NAO
marrom,baixo,180,normal,normal,NAO
verde,alto,30,muito_baixo,longo,NAO
preto,muito_alto,90,bastante_alto,muito_curto,SIM
magenta,alto,150,muito_alto,muito_curto,SIM
marrom,alto,150,alto,longo,NAO
verde,enorme,240,muito_alto,muito_curto,NAO
magenta,bastante_alto,210,nao_assegurado,muito_longo,NAO
azul,enorme,60,bastante_alto,muito_longo,NAO
preto,bastante_alto,90,enorme,normal,NAO
preto,muito_baixo,150,enorme,longo,NAO
verde,enorme,30,bastante_alto,muito_curto,NAO
amarelo,baixo,150,enorme,curto,NÃO
verde,nao_vendavel,210,baixo,muito_longo,NAO
magenta,alto,120,nao_assegurado,muito_curto,SIM
magenta,alto,210,normal,muito_curto,SIM
magenta,alto,30,baixo,muito_curto,SIM
azul,enorme,210,enorme,longo,NÃO
```

Anexo II – Aplicação metodologia DWM utilizando algoritmos K-NN

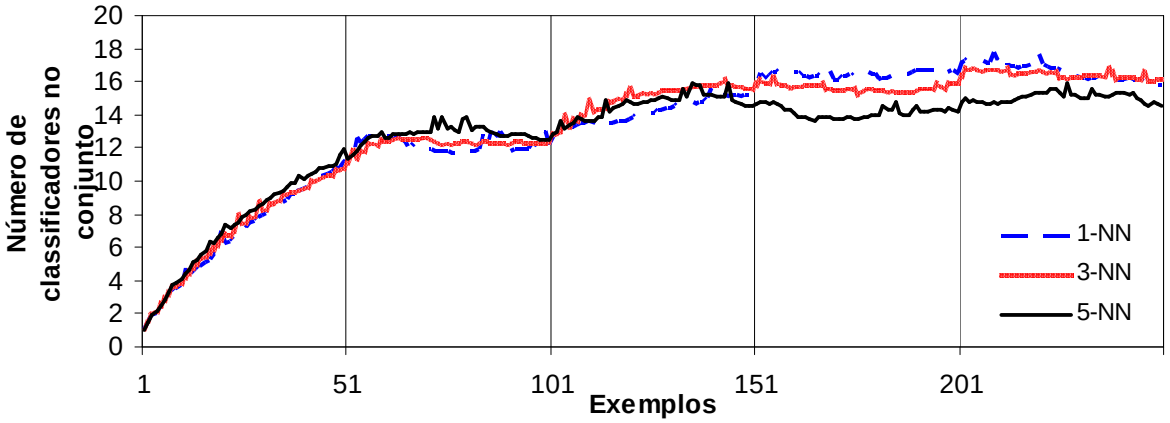


Figura 56 – N° de classificadores - algoritmos K-NN (DWM – Mudança abrupta)

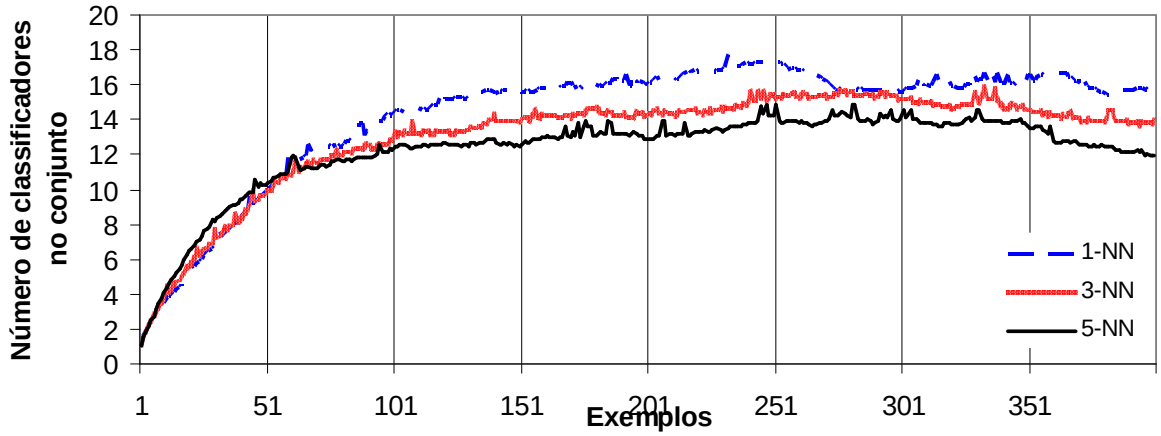


Figura 57 – N° de classificadores - algoritmos K-NN (DWM – Mudança moderada)

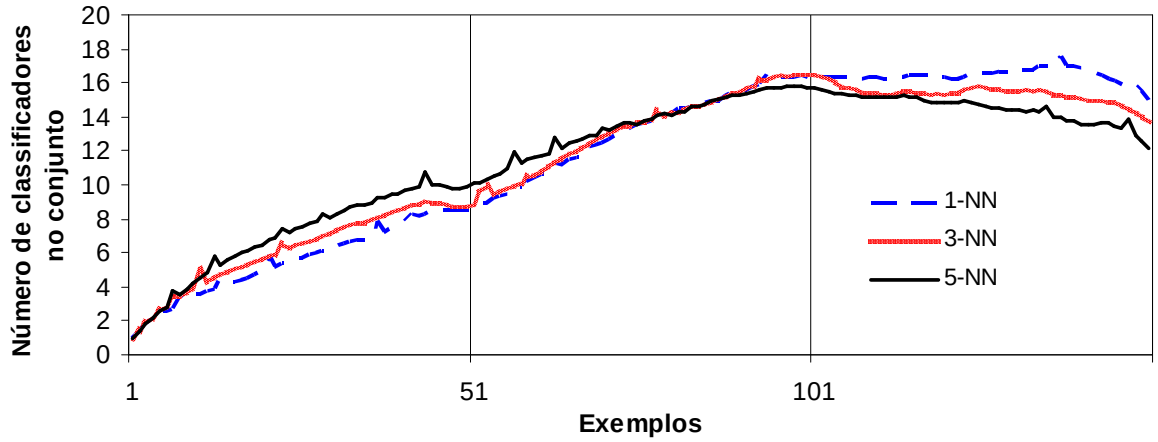


Figura 58 – N° de classificadores - algoritmos K-NN (DWM – Mudança gradual)

ANEXO III - Comparação com diferentes valores de *m* (ligações iniciais) – Mudanças abrupta e gradual

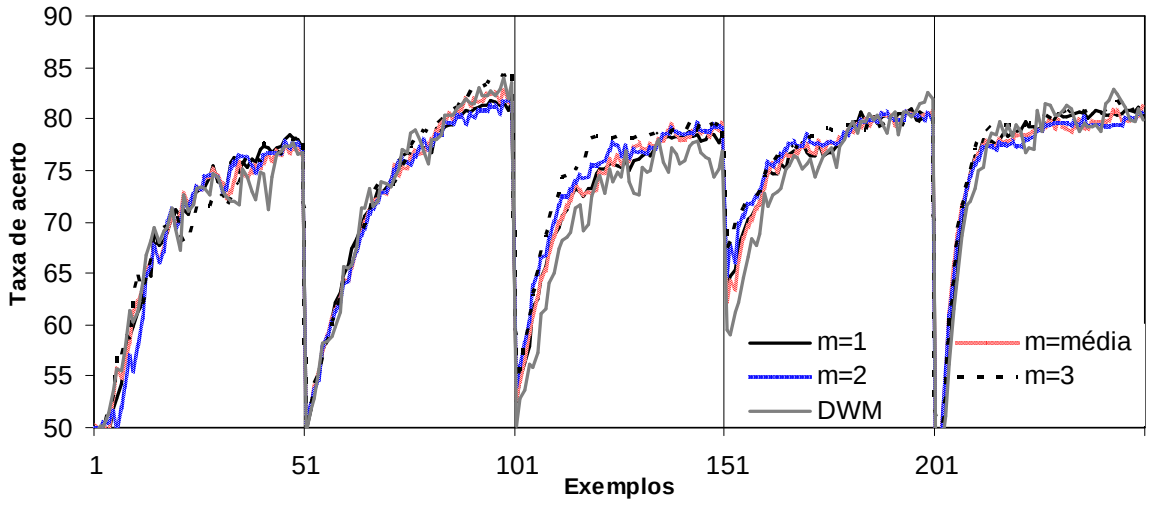


Figura 59 – Taxa de acerto - Mudança Abrupta (3-NN)

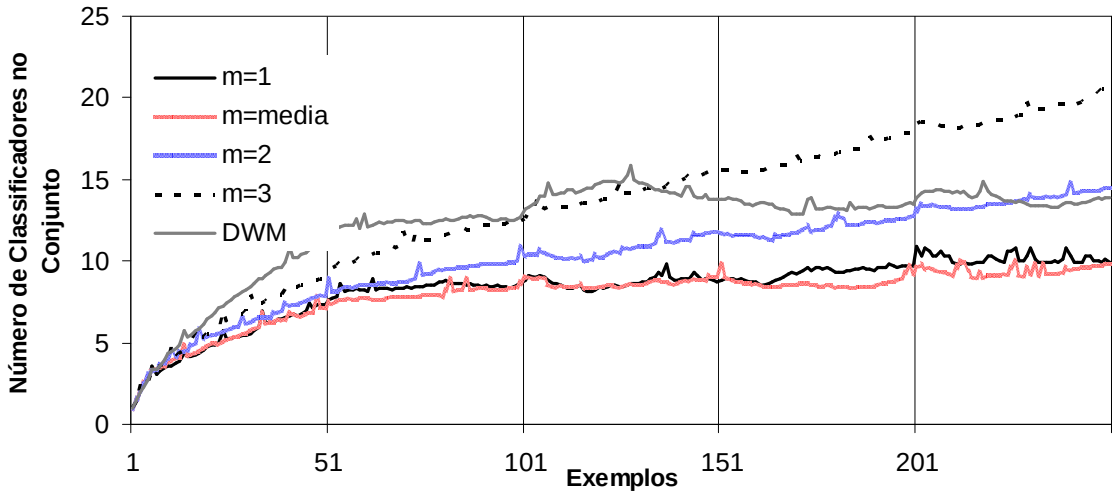


Figura 60 – N° de classificadores - Mudança Abrupta (3-NN)

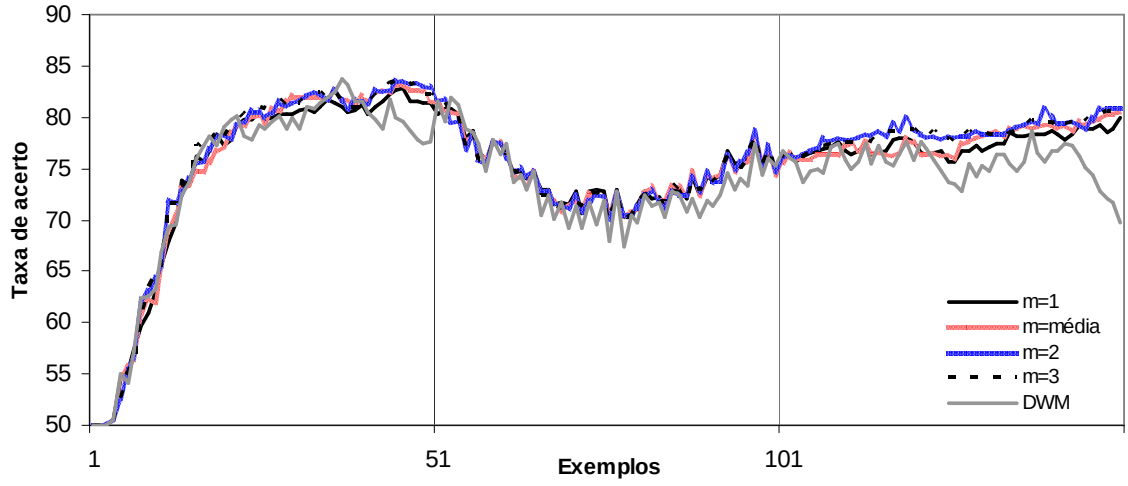


Figura 61 – Taxa de acerto – Mudança Gradual (3-NN)

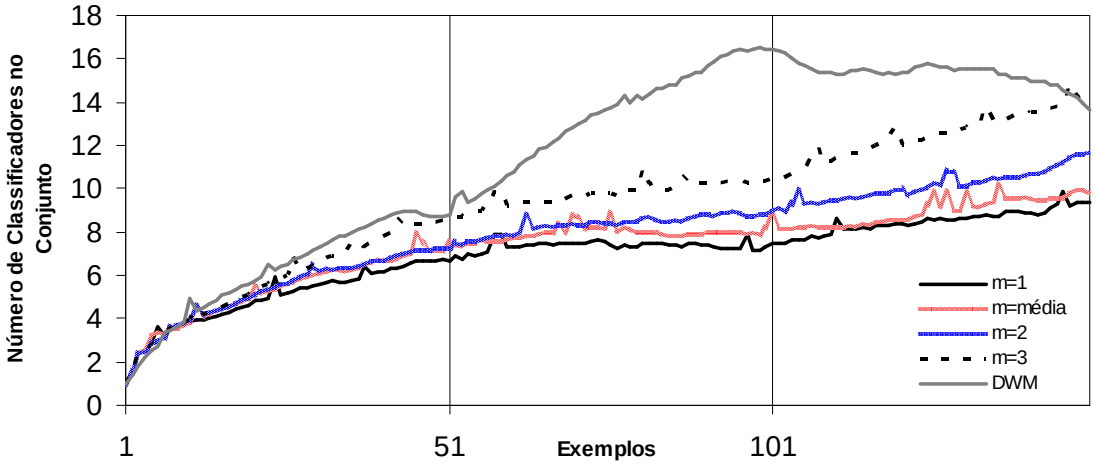


Figura 62 – N° de classificadores - Mudança Gradual (3-NN)

ANEXO IV - *Threshold* para remoção de classificadores – Mudanças abrupta e moderada

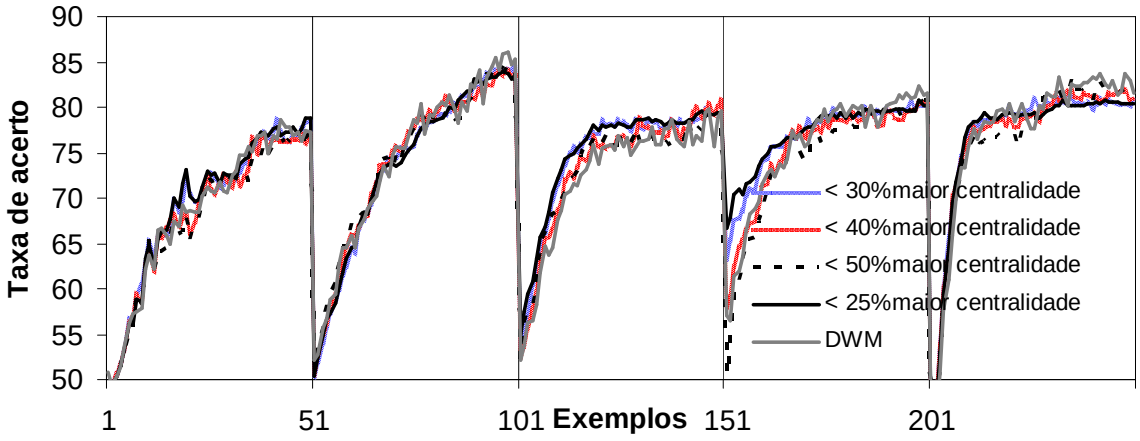


Figura 63 – Taxa de acerto – Mudança Abrupta (3-NN)

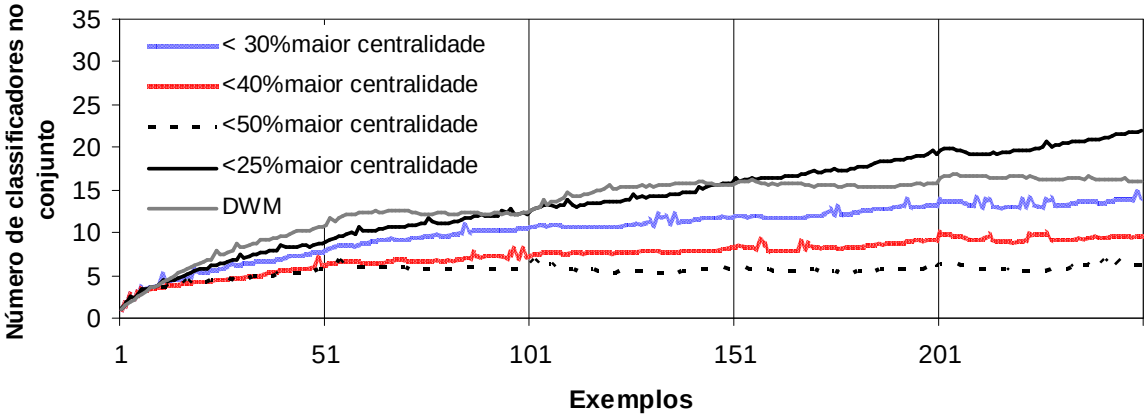


Figura 64 – N° de classificadores – Mudança Abrupta (3-NN)

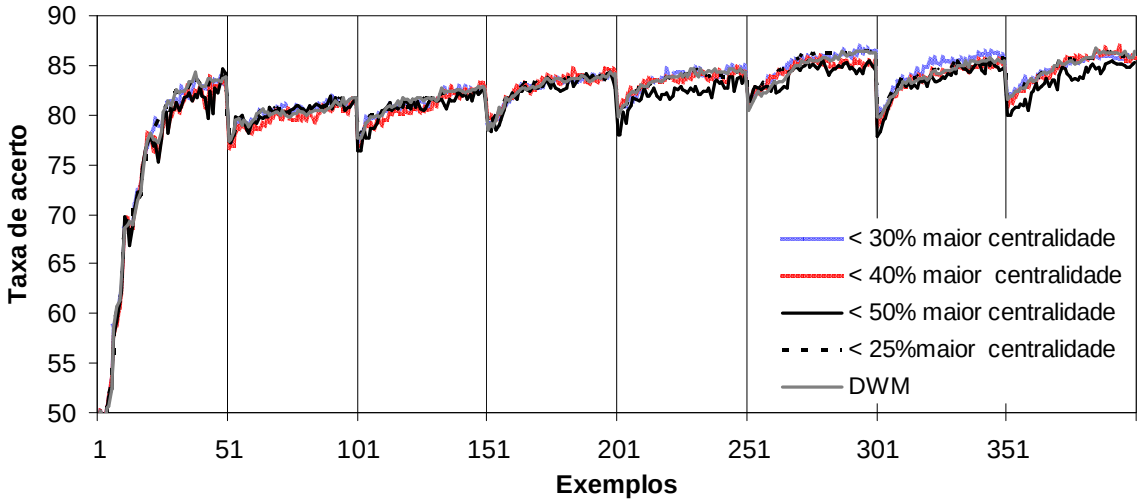


Figura 65 – Taxa de acerto - Mudança Moderada (3-NN)

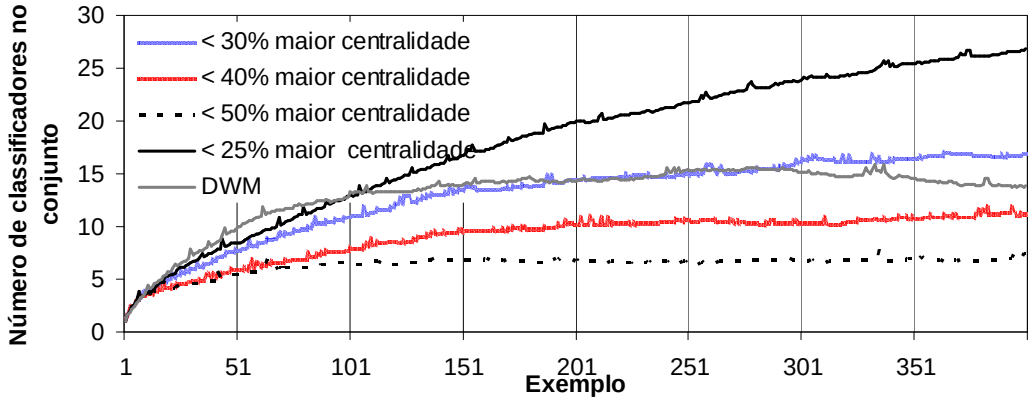


Figura 66 – N° de classificadores – Mudança moderada (3-NN)

ANEXO V - Total de classificadores criados

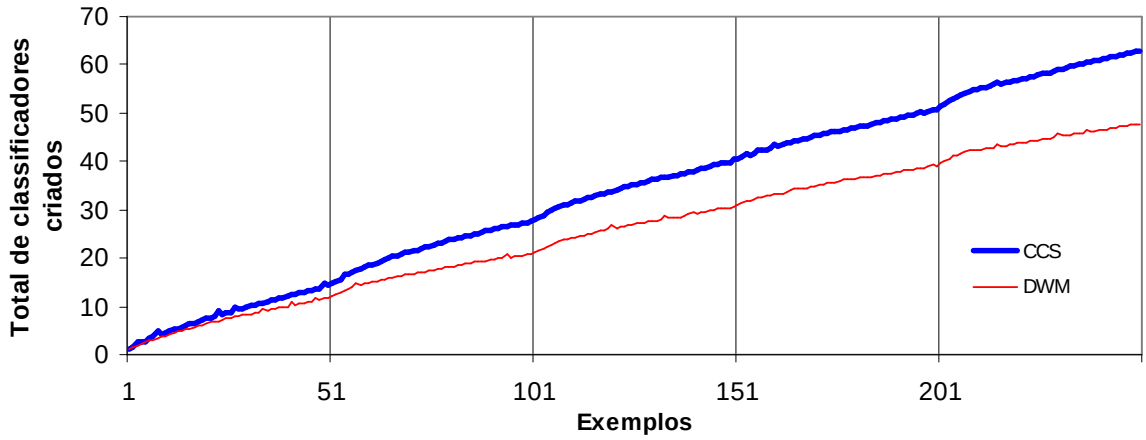


Figura 67 – Total de classificadores criados - Mudança abrupta (3-NN)

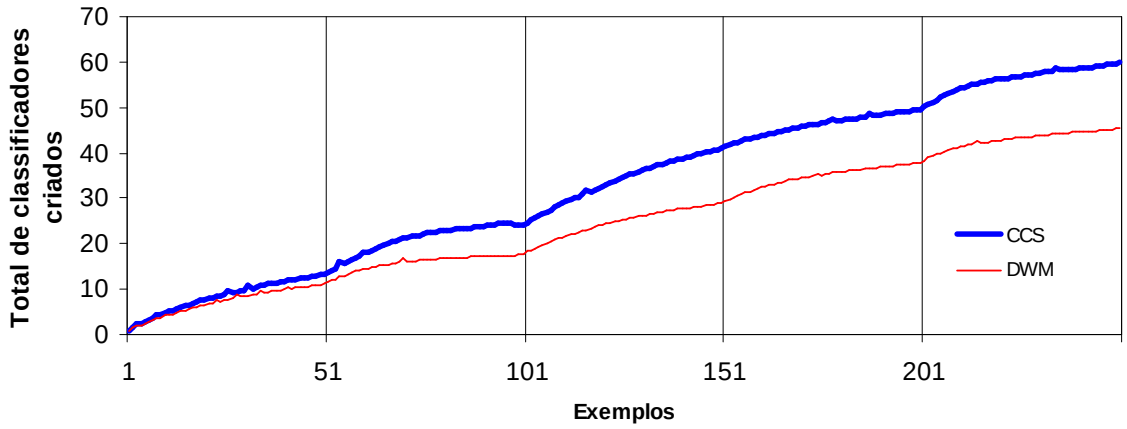


Figura 68 - Total de classificadores criados - Mudança abrupta (J48)

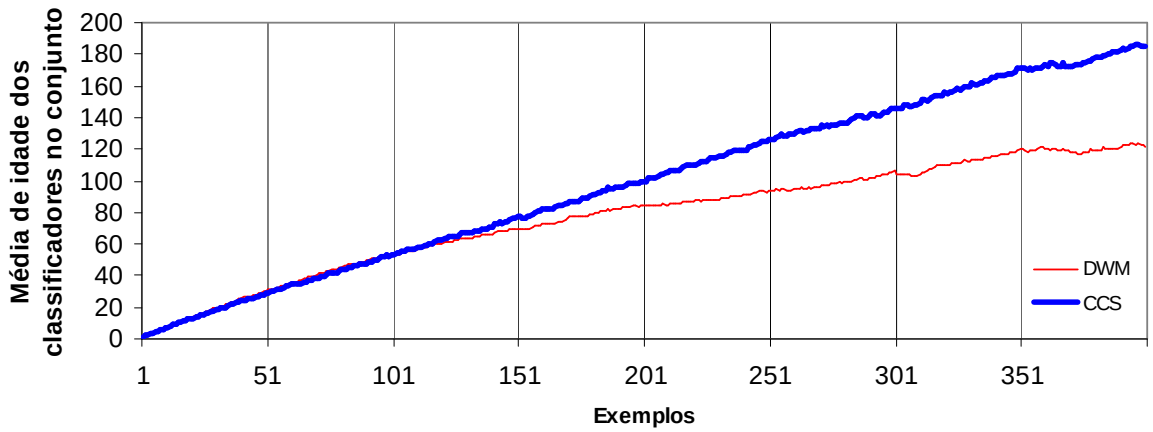


Figura 69 – Total de classificadores criados - Mudança moderada (3-NN)

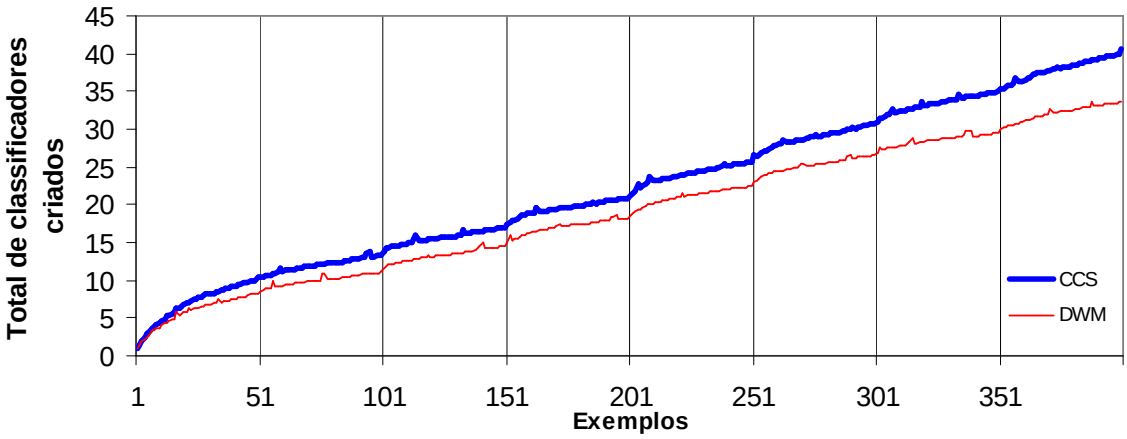


Figura 70 – Total de classificadores criados - Mudança moderada (J48)

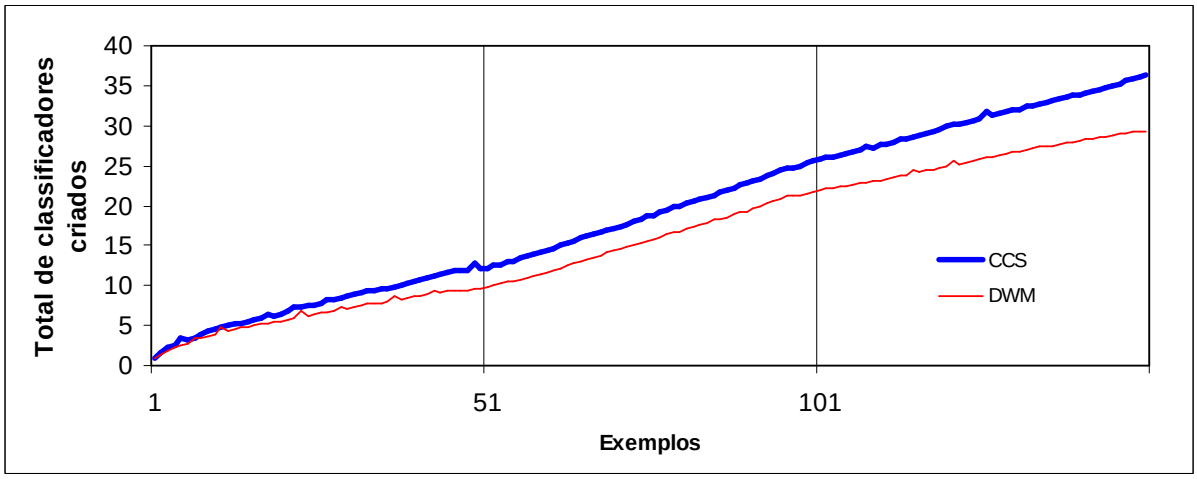


Figura 71 - Total de classificadores criados - Mudança gradual (3-NN)

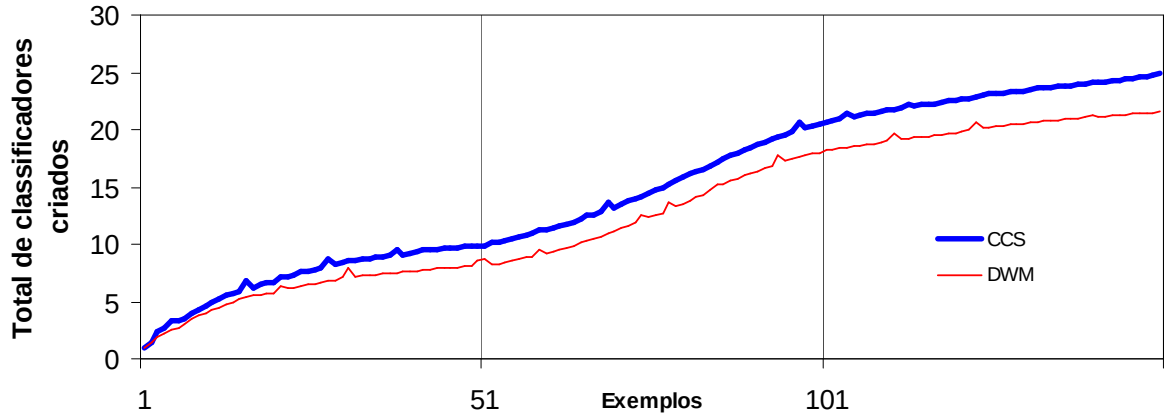


Figura 72 - Total de classificadores criados - Mudança gradual (J48)