Universidade Federal de Minas Gerais
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica - PPGEE

Aprendizado Incremental com Memória Parcial via Grafo de Gabriel

Marcus Vinícius de Freitas Diadelmo

Belo Horizonte - Brasil
2016

DISSERTAÇÃO DE MESTRADO Nº 928

APRENDIZADO INCREMENTAL COM MEMÓRIA PARCIAL VIA GRAFO DE GABRIEL

Marcus Vinícius de Freitas Diadelmo

DATA DA DEFESA: 06/07/2016

Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

APRENDIZADO INCREMENTAL COM MEMÓRIA PARCIAL VIA GRAFO DE GABRIEL

Marcus Vinícius de Freitas Diadelmo

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Orientador: Prof. Cristiano Leite de Castro

Belo Horizonte - MG

Julho de 2016

Diadelmo, Marcus Vinícius de Freitas.
D536a Aprendizado incremental com memória parcial via grafo de Gabriel [manuscrito] / Marcus Vinícius de Freitas Diadelmo. – 2016.
vii, 52 f., enc.: il.

Orientador: Cristiano Leite de Castro.

Dissertação (mestrado) Universidade Federal de Minas Gerais, Escola de Engenharia.

Bibliografia: f. 50-52.

1. Engenharia elétrica - Teses. 2. Teoria dos grafos - Teses.
3. Algoritmos - Teses. I. Castro, Cristiano Leite de.
II. Universidade Federal de Minas Gerais. Escola de Engenharia.
III. Título.

CDU: 621.3(043)

"Aprendizado Incremental Com Memória Parcial Via Grafo de Gabriel"

Marcus Vinícius de Freitas Diadelmo

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica.

Aprovada em 06 de julho de 2016.

Por:

Prof. Dr. Cristiano Leite de Castro
(UFMG) - Orientador

Prof. Dr. Frederico Gadelha Guimarães
DEE (UFMG)

Prof. Dr. Thiago Souza Rodrigues
DECOM (CEFET-MG)

Dr. Luiz Carlos Bambirra Torres
DELT (UFMG)

Dedico este trabalho a todas as pessoas que acreditam que a educação e o estudo são capazes de transformar a sociedade.

Agradecimentos

Primeiramente agradeço a DEUS por me proporcionar força e sabedoria durante toda a caminhada. Somente ele é capaz de saber o tamanho da luta e dedicação investida por mim na busca da realização deste sonho.

Agradeço à minha querida e muito amada Mãe, mulher guerreira e batalhadora que muito me inspirou e apoiou de todas as formas possíveis. Esta vitória é sua.

Agradeço à Maria Gabriela, pelo carinho, amor e companheirismo durante toda a caminhada.

Aos meus irmãos Rosilaine, Renato e Márcio e ao amigo George que sempre me apoiaram nessa jornada e sempre foram grandes incentivadores na minha formação.

Agradeço à minha Tia, Maria Joana (Cota), e à grande amiga, Maria Aparecida (Cida), pessoas como vocês são difíceis de encontrar. Agradeço a Deus por tê-las em minha vida. Ao meu tio, Vagner, agradeço o auxílio e companhia durante a caminhada.

Um agradecimento especial ao Professor Cristiano Leite de Castro pelo convívio e orientação. Seu conhecimento e sua postura foram essenciais na minha formação, você é um exemplo a ser seguido. Ao Dr. Luiz Carlos Bambirra Torres, agradeço pela orientação e conhecimento técnico empregado neste trabalho. Sua orientação foi fundamental para o sucesso deste trabalho.

Agradeço ao Professor Frederico por todo apoio, incentivo e recomendações no início da jornada.

Aos amigos do Instituto Federal de Minas Gerais pelo apoio e compreensão em vários momentos desta jornada.

Agradeço aos professores que tive durante toda a vida, a admiração e respeito que tenho por eles impulsiona o meu desejo de ser um professor cada vez melhor.

Agradeço à FAPEMIG/CAPES pelo apoio financeiro durante a realização deste trabalho.

Resumo

Este trabalho consiste no desenvolvimento de algoritmos de aprendizado incremental com memória parcial, onde a memória parcial é obtida através da estrutura do grafo de Gabriel. São propostas quatro técnicas incrementais, sendo que, em três delas a memória parcial é obtida pelo grafo de Gabriel e pela eliminação de ruídos, e uma quarta técnica que, além de utilizar o grafo de Gabriel e a eliminação de ruídos, utiliza uma medida de afastamento da distribuição real dos dados para selecionar padrões que possam ser relevantes para o aprendizado. Foram realizados testes estatísticos para avaliar os métodos. Nestes testes foram avaliados a equivalência da abordagem incremental e da abordagem tradicional (dados separados em treinamento e teste), comparação entre os algoritmos incrementais desenvolvidos e alguns presentes na literatura. Os resultados comparativos obtidos mostram que as técnicas desenvolvidas são eficientes e possuem a particularidade de não necessitar de especialistas para determinar parâmetros (na maioria das técnicas). Uma breve análise sobre a influência do tamanho da janela de dados indica que, possivelmente, o tamanho da janela não é um parâmetro decisivo para o sucesso do algoritmo.

Abstract

This work presents the development of incremental learning algorithms with partial memory, where the partial memory is obtained by the Gabriel graph structure. Four incremental techniques are proposed. In three of them, the partial memory is obtained by Gabriel graph and noise elimination. Besides considering Gabriel graph and noise elimination, the fourth incremental technique selects relevant patterns from a discrepancy measure of the data true distribution. Statistical tests to evaluated the methods were performed. These tests evaluate the equivalence of the incremental approach with the traditional approach (data separated into training and test), and also the comparison of the incremental algorithms with some others in the literature. The comparative results show that the developed techniques are efficient and have the particularity of not requiring experts to determine the parameters (in most technics). A brief analysis of the influence of the size of the data window indicates that the window size might not be a decisive parameter for the success of the algorithms.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Aprendizados Incrementais são baseados, e caracterizados, em aprendizados adaptativos, progressivos e dinâmicos. É a forma de aprendizado utilizada pelos humanos, onde o aprendizado é adquirido e aprimorado com o passar do tempo, ou seja, o conhecimento é formado a partir de um conjunto de informações e, à medida que novas informações são adquiridas, o conhecimento é aprimorado.

De acordo com Fisher (1987) o aprendizado humano pode ser classificado como um processo gradativo de formação de conceitos, adaptando novos conhecimentos aos já adquiridos, de forma a torná-lo mais sólido, conciso e, portanto, melhor.

Segundo Yoshida (2007) o aprendizado humano pode ser caracterizado como incremental, uma vez que os conceitos são formados ao longo do tempo através da aquisição de novas informações. Yoshida (2007) defende que o aprendizado de máquina, diferentemente do aprendizado humano, é melhor se feito com toda a informação disponível, dado que os processos e/ou algoritmos tradicionais apresentam um bom desempenho quando possuem toda a informação disponível.

Na literatura recente são encontrados trabalhos que tratam da detecção e do tratamento de mudança de conceito (*concept drift*) (GAMA et al., 2014). Este tipo de análise tem o objetivo de verificar a mudança, ou não, de características na distribuição de dados. Essa análise torna-se importante para determinar momentos em que o aprendizado (ou modelo preditor) está obsoleto. Gama et al. (2014) destaca alguns tipos de mudança, descritas a seguir e mostradas na Figura 1.

- *Sudden Drift* – quando a distribuição se altera de forma brusca.
- *Gradual Drift* – dados gerados por duas fontes distintas por um período e em determinado tempo, a razão da geração dos dados se altera.
- *Incremental Drift* – ocorre mudanças suaves na distribuição.
- *Recurring concepts* – fontes distintas geram os dados alternadamente em períodos irregulares.

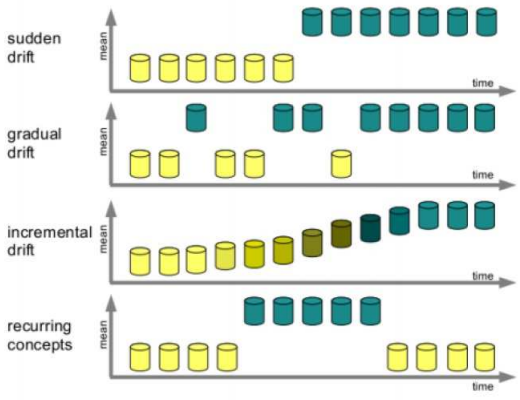

Figura 1 – Tipos de Mudança em Distribuição. Retirado de (GAMA et al., 2014)

Além das razões para a utilização de algoritmos incrementais descritas acima, Zheng et al. (2010) destaca outros motivos para a utilização destes algoritmos:

- A utilização de algoritmos incrementais são aplicadas em problemas com restrição de memória e/ou com baixo desempenho computacional.
- Os algoritmos incrementais são essenciais em problemas onde a obtenção de dados acontece de forma temporal, sendo necessário a atualização do modelo a partir da chegada de novos dados.

Existe uma gama de problemas onde os algoritmos tradicionais não são aplicáveis. Dessa forma, a utilização de algoritmos incrementais se torna uma excelente opção. Podemos destacar, principalmente dois cenários reais. O primeiro cenário consiste em problemas *big data*, onde é inviável utilizar um algoritmo preditor capaz de processar grande volume de dados (AGRAWAL; DAS; ABBADI, 2011). Neste caso, um algoritmo incremental particiona esse grande volume de dados e processa os mesmos por etapas. O segundo cenário representa os problemas de *data stream*, onde os dados chegam em tempos diferentes. Neste cenário, um algoritmo tradicional necessita esperar a chegada dos dados e os acumular para gerar o modelo preditor, essa tarefa se torna inviável, pois com o passar do tempo o volume de dados cresce tornando impraticável o treinamento(LASKOV et al., 2006). Um algoritmo incremental, para essa situação, tem a função de selecionar somente os padrões relevantes e eliminar os demais, poupando memória e tornando possível o tratamento deste tipo de problema.

1.1 Aprendizado Incremental

Aprendizado de Máquina aplicado em problemas reais possui o desafio de manter permanentemente um modelo preditor com um alto índice de acerto, pois estes problemas apresentam mudanças de dinâmica, mudança de distribuição, grande volume de

informações, tornando difícil a tarefa de encontrar um único preditor capaz de lidar com essas características. Dessa forma surge o Aprendizado Incremental.

Algoritmos Incrementais devem ser capazes de atualizar o modelo, se necessário, sempre que novos conjuntos de dados estão disponíveis. Gama et al. (2014) destaca que, além de atualizar o modelo, estes algoritmos devem ser capazes de esquecer informações (dados) consideradas irrelevantes.

Gama et al. (2014) destaca, ainda que, algoritmos de aprendizagem muitas vezes precisam operar em ambientes dinâmicos, que podem mudar inesperadamente. Uma propriedade desejável desses algoritmos é a sua capacidade de incorporar novos dados. Se o processo de geração de dados não é estritamente estacionário (como acontece com a maioria das aplicações do mundo real), o conceito, que estão prevendo, pode mudar ao longo do tempo. A capacidade para se adaptar a tais mudanças pode ser visto como uma extensão natural para sistemas de aprendizagem incremental (GIRAUD-CARRIER, 2000), os quais aprendem modelos preditivos exemplo, por exemplo ou lote por lote.

Gama et al. (2014) aponta que o processo de aprendizado adaptativo possui as seguintes etapas:

- Predição. Dado novos exemplos, determina-se a saída utilizando o modelo atual;
- Diagnóstico. Após um tempo, os rótulos reais dos dados são obtidos e então torna-se possível estimar o erro de classificação;
- Atualização. Utiliza-se os novos exemplo, rotulados, para atualizar o modelo preditor.

A Figura 2 mostra um esquema genérico do aprendizado adaptativo.

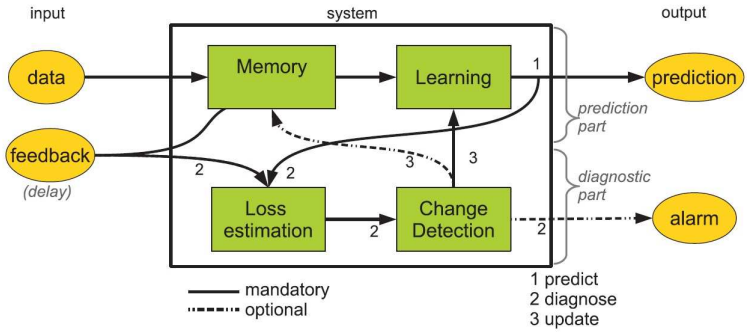

Figura 2 – Esquema genérico do Aprendizado adaptativo (GAMA et al., 2014).

Na Figura 2 vemos que os dados são armazenados em memória e transferidos ao modelo treinado, este, por sua vez, deve informar a saída predita. O que torna o algoritmo incremental é o monitoramento da saída predita, tornando possível medidas temporais da eficiência do modelo, e se possível gerar um novo. O momento de atualizar o modelo indica

o tipo de treinamento utilizado. De acordo com Gama et al. (2014) pode-se destacar algumas metodologias, como *Single Example*, onde a cada novo dado o algoritmo é re-treinado, alguns autores denominam esta técnica de treinamento online. Existe a técnica de *Multiple Examples*, onde é utilizado um conjunto de dados recentes, janelas, para re-treinar o modelo, podendo este conjunto ser de tamanho fixo ou não. A técnica de memória parcial consiste em armazenar dados representativos e utilizá-los posteriormente para gerar um novo modelo.

O aprendizado adaptativo é uma forma geral de apresentar algoritmos que trabalham em ambientes dinâmicos. Dentro dessa definição, podemos destacar dois principais métodos. Aprendizado online e Aprendizado incremental. No primeiro, a cada novo exemplo é realizado a predição com o modelo recorrente, caso a predição esteja incorreta, o modelo é alterado. O segundo utiliza cada novo exemplo, ou lote de exemplos para atualizar o modelo, juntamente com as informações antigas (GAMA et al., 2014).

Na literatura são encontradas definições distintas de aprendizado online e incremental, embora alguns autores considerem estes termos equivalentes.

No trabalho de Pinto (2005) é destacado que um aprendizado é dito incremental quando é capaz de atualizar um modelo a partir de novos dados disponíveis, já o aprendizado online é capaz de receber um novo exemplo, atualizar o modelo e detectar se tal mudança está correta. Em outras palavras, o aprendizado online atualiza o modelo a partir de um novo e único dado, através da detecção do erro e o aprendizado incremental atualiza o modelo a partir de um novo conjunto de dados.

Gama et al. (2014) destaca, ainda, que o aprendizado online atualiza o modelo com o dado mais recente. Este processo é guiado pelo erro, ou seja, o modelo se atualiza dependendo da classificação do exemplo atual. Já os algoritmos incrementais processam os dados de entrada, lote por lote (ou um por um), e atualizam o modelo depois de receber cada lote de dados. Estes algoritmos tem acesso a exemplos ou resumos de exemplos anteriores.

Syed, Liu e Sung (1999) aponta que os algoritmos adaptativos são utilizados para processar dados em grandes escalas, onde não é possível processar todos os dados de uma vez, particionando a base de dados e armazenando as informações importantes.

No trabalho de Polikar et al. (2001) é destacado que um algoritmo, para ser considerado incremental, deve seguir os seguintes critérios:

- Capacidade de adquirir novos conhecimentos a partir da informação de novos dados.
- Não deve recorrer ao conjunto de dados original, a fim de atualizar o modelo.
- Preservar conhecimentos anteriores.
- Acomodar novas classes a partir da introdução de novos dados.

Além dos critérios descritos por Polikar et al. (2001), pode-se destacar que o aprendizado incremental não precisa se basear em erros de classificação para selecionar padrões relevantes. Caso o aprendizado incremental se baseie em erros de predição, este pode ser classificado como um aprendizado online.

Em seu trabalho, Ade e Deshmukh (2013) apresenta dois métodos tradicionais de aprendizado incremental, estes métodos são mostrados na Figura 3

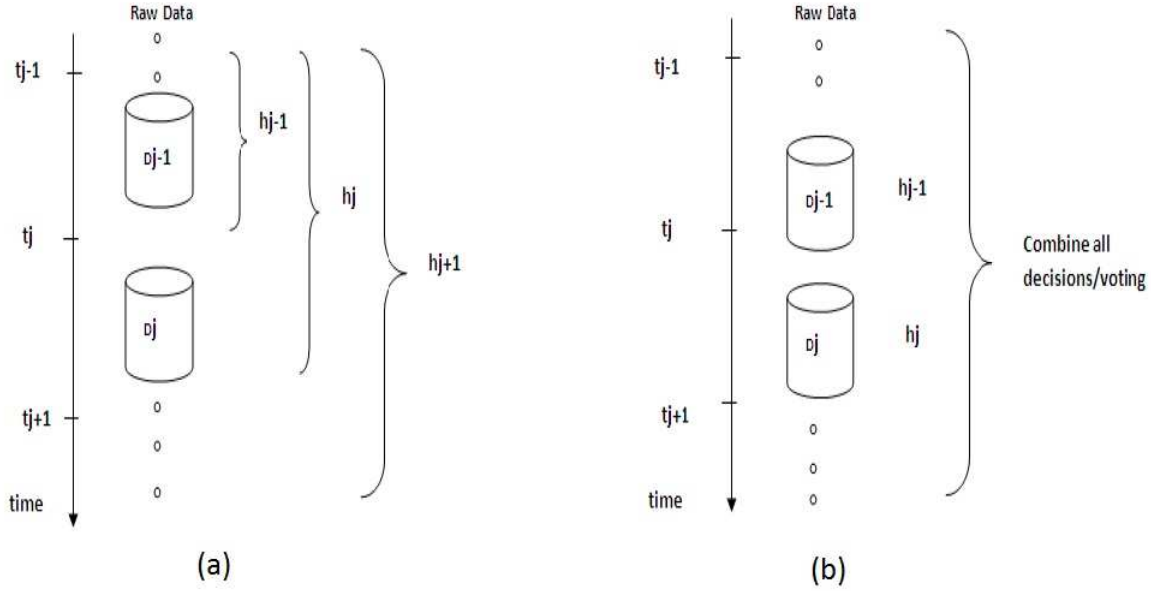

Figura 3 – Formas tradicionais de aprendizado incremental (a). Acumulação de Dados (b). Adaptado de (ADE; DESHMUKH, 2013)

No primeiro método, um novo conjunto de dados D_j é recebido e, então agrupado com a hipótese $hj - 1$ para gerar uma nova hipótese hj.

No segundo método, um novo conjunto de dados D_j gera uma hipótese simples. A hipótese final é obtida através da combinação, por algum mecanismo de votação, das hipóteses simples já obtidas.

1.1.1 Memória Parcial

Aprendizados de memória parcial são sistemas incrementais que selecionam e mantêm uma parte dos exemplos de treinamento passados e utilizam, em conjunto com novos exemplos, para formar novos modelos. Maloof e Michalski (2000) destaca que tais sistemas podem ser menos susceptíveis ao *overtraining* em problemas com mudança de distribuição, se comparados a outras técnicas.

A principal ideia acerca da memória parcial, consiste em guardar somente informações relevantes em memória. O ponto chave consiste em como o sistema irá selecionar estes dados relevantes. A escolha dos dados acarretará na precisão, capacidade de memória e habilidade em lidar com mudanças de distribuição.

1.1.2 Objetivo

Neste trabalho consideramos um cenário de *data stream*, onde os dados chegam temporalmente em janelas, ou seja, blocos de dados chegam a cada instante de tempo t. Os rótulos dos dados chegam com certo atraso, sendo necessário, primeiramente, apresentá-los ao modelo atual para classificação e somente depois utilizá-los para compor o conjunto de treinamento.

A Figura 4 mostra o cenário em que o trabalho se apoia. Os pacotes de dados chegam de forma separada e em tempos distintos, além do atraso no recebimento dos rótulos dos dados. Com isso, é necessário obter um classificador que, antes de receber os rótulos, que podem atrasar, realize a predição dos dados. No instante inicial, quando não existe um classificador, é necessário esperar os rótulos dos dados para compor o treinamento e gerar o primeiro classificador.

Figura 4 – Cenário considerado no trabalho.

A obtenção de classificadores incrementais se baseia na realidade do problema. No cenário considerado, faz-se necessário trabalhar de forma incremental, uma vez que o fluxo de dados é contínuo, podendo estourar a capacidade dos recursos computacionais (hardware e software) envolvidos e inviabilizar o treinamento de um grande volume de dados. Além disso, existe a necessidade de obter classificadores a qualquer momento em que se deseja classificar novos dados. Apesar de considerarmos um cenário de *data-stream*, os algoritmos desenvolvidos também podem ser aplicados no aprendizado de grandes volumes de dados. Para isso, basta realizar a divisão dos dados em lotes, neste caso a componente temporal é abstraída.

O classificador para este cenário deve possuir estratégias de esquecimento, no intuito de poupar memória e facilitar o treinamento.

O objetivo do trabalho consiste em trabalhar com problemas com as características citadas acima. A proposta dessa dissertação está no desenvolvimento de classificadores incrementais que utilizem mecanismos de esquecimento e armazenamento para seleção de dados. Estes mecanismos baseiam-se em informações geométricas da distribuição espacial dos dados.

1.1.3 Organização do Trabalho

O restante do trabalho se apresenta da seguinte maneira:

No Capítulo 2 são apresentadas algumas técnicas presentes na literatura bem como uma análise sobre as principais contribuições do trabalho. O Capítulo 3 traz uma avaliação sobre o grafos de proximidade com o intuito de definir o processo de construção do Grafo de Gabriel e como se utiliza as informações do Grafo de Gabriel para construção de filtros e de um classificador de margem larga. O Capítulo 4 descreve as metodologias implementadas neste trabalho, apresentando diagramas e comparações entre elas. O Capítulo 5 descreve os processos de avaliação das metodologias, através de testes estatísticos de comparação de resultados. Por fim o Capítulo 6 mostra as conclusões acerca dos resultados obtidos. Uma análise sobre a influência do tamanho da janela é apresentada neste capítulo. Também são mostradas algumas propostas de continuidade do trabalho.

1.1.4 Considerações

Com as definições descritas nas seções anteriores, é possível fazer algumas considerações que serão utilizadas no decorrer do trabalho. Os algoritmos desenvolvidos são considerados incrementais, por apoiarem nos requisitos descritos em (POLIKAR et al., 2001) . Além disso, após verificar as considerações da literatura, ao diferenciar algoritmos incrementais e *onlines*, e como as técnicas desenvolvidas não se baseiam em erros para atualizar o modelo, não é possível considerar que as técnicas sejam baseadas em aprendizado *online*. O modelo será atualizado sempre que novos lotes de dados forem apresentados ao sistema.

As técnicas implementadas utilizam mecanismos de seleção e esquecimento de dados e armazenamento em memória. Com isso destaca-se que os algoritmos desenvolvidos neste trabalho são classificados como Algoritmos Incrementais com memória parcial.

2 Algoritmos de Aprendizado Incremental

Neste Capítulo são apresentadas algumas técnicas de Aprendizado Incremental presentes na Literatura e que serviram de influência para o desenvolvimento do Trabalho.

2.1 Aprendizado Incremental com SVM

Através da utilização de características importantes disponibilizada pelo treinamento de uma SVM, Syed, Liu e Sung (1999) propõe um mecanismo de filtragem, onde os dados são divididos em janelas e em cada janela são selecionados os vetores de suporte. A união dos vetores de suporte de todas as janelas formam a base de dados que será utilizada para obter o classificador final. Este processo é mostrado na Figura 5, onde *Initial Empty Concept* implica que inicialmente não existe nenhum modelo preditor. *Target Concept 1* é o modelo obtido pelo treinamento da SVM para o *subset 1*. *Target Concept 2* é o modelo obtido pelo treinamento da SVM para o *subset 2* juntamente com os *Suport Vectors* do *Target Concept 1* e assim sucessivamente até a obtenção do *Final Concept*.

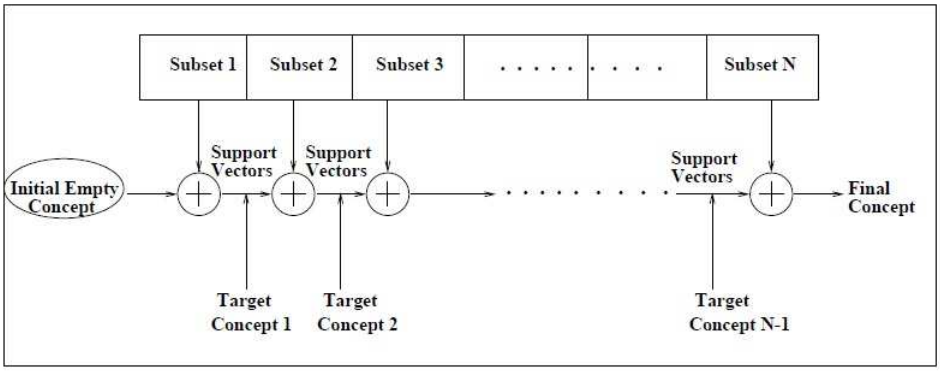

Figura 5 – Técnica Incremental utilizada por (SYED; LIU; SUNG, 1999)

A técnica apresentada por Syed, Liu e Sung (1999) possui algumas características importantes a serem destacadas. Como pontos positivos está o fato do treinamento de uma SVM retornar os vetores de suporte que, por si só, é uma forma de filtragem, uma vez que ao substituir todos os dados pelos vetores de suporte, o treinamento retornará o mesmo hiperplano separador.

Da mesma forma, podemos destacar como ponto negativo a perda de determinadas informações através do janelamento. Para exemplificar, consideremos um dado que no treinamento com todos os dados se torna um vetor de suporte, porém, quando entra em uma janela com uma distribuição diferente se torna um dado irrelevante, naquela janela, e então é descartado de forma definitiva.

Outro ponto negativo desta técnica está no fato de a cada janela necessitar de um treinamento de uma SVM, bem como a sintonização dos parâmetros deste treinamento, tornando o processo complexo

2.2 Algoritmo *Learn++*

O Algoritmo *Learn++* foi desenvolvido por Polikar et al. (2001) e inspirado no **AdaBoost** (SCHAPIRE, 1999). Este algoritmo utiliza o conceito de que a união de diversos classificadores "fracos", pode gerar um classificador "forte". A ideia está em particionar o conjunto de entrada e para cada partição gerar uma classificador "fraco". Após a obtenção dos classificadores "fracos"é utilizada uma votação através dos pesos (*weighted majority voting*) para combinar os classificadores e então obter o classificador "forte".

Neste algoritmo os dados são divididos em m janelas, sendo que em cada janela a distribuição dos dados receberá um peso inicial. Na sequência, a primeira janela é selecionada e subdividida em conjunto de treinamento e teste de forma aleatória. Então é gerado uma hipótese com um classificador "fraco"e calcula-se o erro de classificação com os dados de testes. Caso o erro possua valor maior ou igual a $0,5$ a hipótese é descartada e uma nova partição dos dados em treinamento e teste é feita, isso ocorre até que o erro alcance um valor menor que $0,5$. Após, é calculado o erro normalizado e a hipótese é combinada através do *weighted majority voting* e então é obtido a hipótese combinada. Com a hipótese combinada obtida, realiza-se o cálculo do erro e a atualização dos pesos. Este processo se repete até que todas as janelas tenham sido treinadas. Finalmente a hipótese final é obtida pelo processo de *weighted majority voting*.

Esta técnica apresenta como característica positiva o fato do treinamento dos classificadores "fracos"não utilizar uma forma complexa de treinamento.

O ponto negativo, assim como no método anterior, está na perda de informações importantes no processo de partição e treinamento de janelas individualmente.

2.3 Aprendizado Incremental com *Prototypes* e SVM

No trabalho de Zheng et al. (2010) foi proposta uma técnica utilizando *prototypes* para realizar um primeiro processo de filtragem e então um classificador do tipo SVM realiza uma segunda filtragem. Os *Prototypes* são formados utilizando uma estrutura de vizinhança, onde os dados de entrada são comparados com seus vizinhos e, através da medida de alguns parâmetros, como a distância, define se este dado deve ou não ser um *prototype*.

A formação dos *prototypes* ocorre da seguinte maneira. Dado um novo exemplo (x_i), este será adicionado ao conjunto de *prototypes* caso não exista nenhum outro exemplo da mesma classe. Caso exista dados de mesma classe é necessário encontrar os dois dados mais próximos de x_i(*"Wwinner"e "Wrunner-up"*) e calcular a distância de cada um deles com relação a x_i. Caso a distância entre x_i e *"Wwinner"* supere um limiar, que se adapta de forma automática, ou a distância entre x_i e *"Wrunner-up"* supere outro limiar, que também se adapta de forma automática. x_i é adicionado ao conjunto de *prototypes*. Em outras palavras, um exemplo é considerado um *prototypes* caso não possua nenhum outro dado de sua classe ou este dado esteja distante dos demais dados.

Após a seleção dos *prototypes* um classificador SVM seleciona os Vetores de Suporte do conjunto de *Prototypes* e os reinsere no conjunto. A Figura 6 mostra como ocorre este processo, os *Prototypes* (LPs) são selecionados a partir do conjunto de entrada e os vetores de suporte (LSVs) são selecionados a partir do conjunto de *Prototypes*.

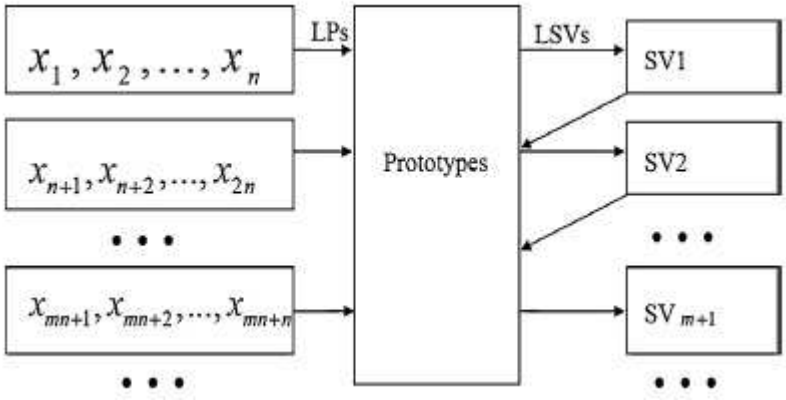

Figura 6 – Técnica Incremental utilizada por (ZHENG et al., 2010)

2.4 Considerações

As técnicas descritas acima apresentam características importantes que os tornam bons classificadores incrementais, contudo, o sucesso destes algoritmos passa por atualização de parâmetros. A principal contribuição das técnicas desenvolvidas neste trabalho consiste em não necessitar de especialistas para definir parâmetros. Sendo que em três das

técnicas implementadas não é necessário nenhum parâmetro. A quarta técnica apresenta dois parâmetros que necessitam ser definidos a priori. Estes parâmetros estão relacionados principalmente com a capacidade de memória das máquinas, onde os algoritmos serão utilizados. Estes parâmetros não influenciam, de forma significativa, no sucesso dos algoritmos. Dessa forma, ressalta-se que as técnicas implementadas independem do conhecimento do usuário para que se obtenham bons resultados.

3 Classificador de Margem Larga

Este capítulo apresenta as ferramentas utilizadas para implementação de um classificador de margem larga através da estrutura de um grafo.

O decisor incremental utilizado neste trabalho se baseia em técnicas da teoria dos grafos, mais especificamente no Grafo de Gabriel (GG). As subseções seguintes são destinadas a apresentar a teoria de construção e extração de informações importantes do GG para implementação do decisor.

Antes de iniciarmos a discussão sobre o GG, definiremos o que é um grafo. De acordo com BARROSO (2007) um grafo é uma estrutura matemática constituída de dois conjuntos, sendo um V, finito e não vazio, de n vértices, e outro E, de m arestas, que são pares não ordenados de elementos de V. De forma geométrica, em um grafo os pontos são associados aos vértices e as linhas que ligam os vértices são chamadas de arestas. A Figura 7 mostra um grafo $G(V, E)$, sendo que $V = 1, 2, 3, 4, 5$ e E=$(1, 3), (2, 3), (2, 4), (3, 4), (4, 5)$. Os vértices interligados por uma aresta são chamados de vértices adjacentes.

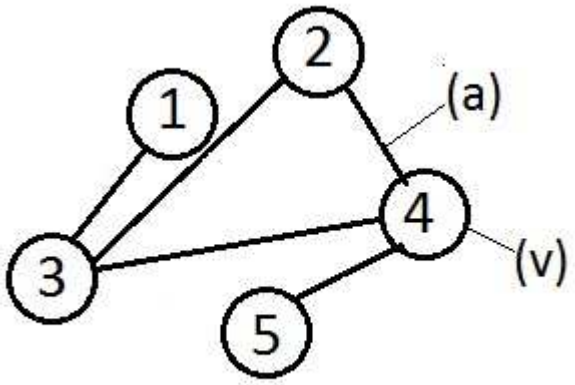

Figura 7 – exemplo de um grafo arbitrário.

3.1 Diagrama de *Voronoi*

De acordo com Okabe, Boots e Sugihara (1992) o diagrama de *voronoi* captura informações de proximidade de um conjunto de pontos a partir da decomposição do plano em polígonos convexos, onde um polígono convexo que engloba um ponto p_i contém a região onde nenhum outro ponto do plano possui uma distância menor.

Dessa maneira, temos que a região de *voronoi* para um ponto p_i é dada por:

$$Vor(p_i) = \left\{x \in R^2 | d(x, p_i) \leq d(x, p_j) \forall j \neq i\right\}$$

sendo $d(.)$ o operado que calcula a distância entre dois pontos. A Figura 8 mostra o diagrama de *voronoi* para um determinado conjunto P.

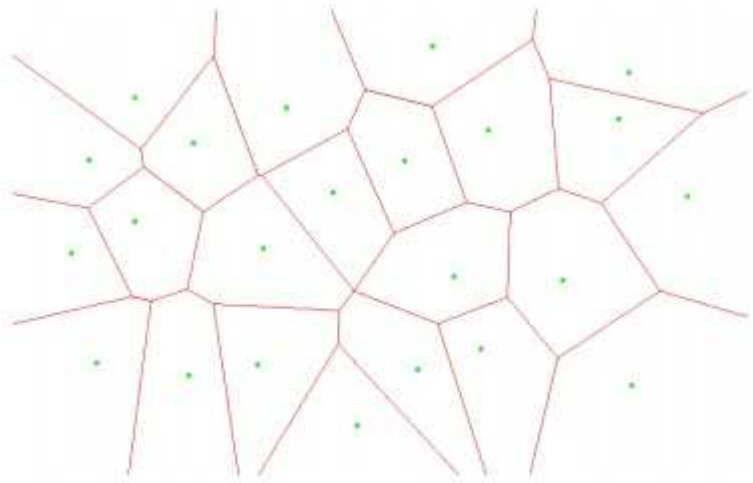

Figura 8 – Diagrama de *Voronoi*.

3.2 Triangulação de *Delaunay*

De acordo com Duran e Perez (2015) uma triangulação de um conjunto P de n pontos é um grafo retilíneo, plano com vértices em P e com um conjunto máximo de arestas. A triangulação de *Delaunay* de um dado conjunto P, segundo Li e Kuo (1998a) é o grafo dual do Diagrama de *voronoi* deste mesmo conjunto P, como mostra a Figura 9. Torres (2012) destaca que a Triangulação de *Delaunay* pode ser modelada por um grafo planar G_p, onde uma aresta interligando quaisquer dois vértices é definida se e somente se existir um círculo que contenha somente estes dois vértices.

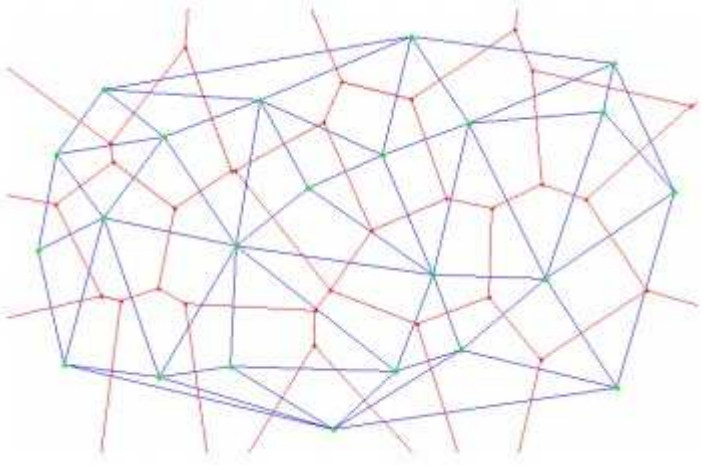

Figura 9 – Triangulação de *Delaunay* e Diagrama de *Voronoi*.

3.3 Grafo de Gabriel

Segundo Duran e Perez (2015) o Grafo de Gabriel G_G de um determinado conjunto de pontos P, é um subgrafo da triangulação de *Delaunay* de P. Uma determinada aresta pertence ao G_G se e somente se não existir nenhum outro ponto dentro de uma circunferência que tangencie os dois vértices da aresta considerada. Berg et al. (2008) destaca que o G_G é um grafo conexo.

3.3.1 Definição

Seja o conjunto de dados $K = \{(\mathbf{x}_i, \mathbf{y}_i) \mid i = 1 \dots N\}$, onde $\mathbf{y}_i \in \{+1, -1\}$ e $\mathbf{x}_i \in R^n$, o G_G de K possui um conjunto de vértices $V = \{\mathbf{x}_i \in K | i = 1, 2...N\}$ e um conjunto de arestas A, que são pares de vértices não orientados pertencentes a V. Uma aresta será formada entre os vértices $\mathbf{x}_i$ e $\mathbf{x}_j$ se, e somente se,

$$\delta^2(\mathbf{x}_i, \mathbf{x}_j) \leq \left[\delta^2(\mathbf{x}_i, \mathbf{x}_z) + \delta^2(\mathbf{x}_j, \mathbf{x}_z)\right], \tag{3.1}$$

$\forall\, \mathbf{x}_z \in V$ e $i \neq j \neq z$, onde $\delta(\cdot)$ é o operador que calcula a distância Euclidiana entre os vértices.

A Figura 10 mostra a construção do Grafo de Gabriel para um determinado conjunto de dados.

Foi mostrado em Torres (2012) que a obtenção do Grafo de Gabriel fornece uma importante informação sobre a margem de separação das classes, uma vez que amostras de classes distintas interligadas por uma aresta do GG são amostras pertencentes às bordas de separação, tais amostras são chamadas de Vetores Geométricos. A utilização dos Vetores Geométricos proporciona informações importantes para obtenção de um classificador de Margem Larga. Formas de obter classificadores de margem larga utilizando os Vetores Geométricos serão descritos no capítulo seguinte.

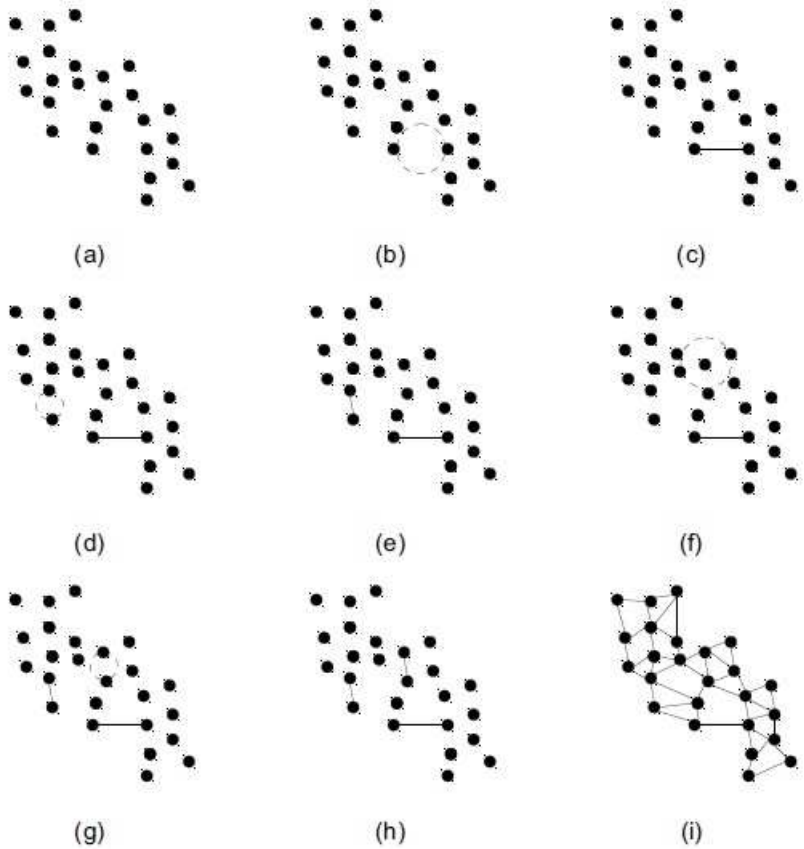

Figura 10 – Construção do Grafo de Gabriel. Adaptado de (TORRES, 2012)

3.4 Classificador

3.4.1 Vetores Geométricos

A partir da construção do Grafo de Gabriel, pode-se utilizar algumas técnicas, presentes na literatura, para obtenção de classificadores de margem larga. Para tal, é necessário encontrar primeiramente os dados presentes na região de separação entre as classes (TORRES; CASTRO; BRAGA, 2015). Os dados presentes na região de separação são chamados de Vetores Geométricos (VG) e são definidos como sendo os vértices de classes distintas que são ligados por uma aresta, como pode ser visto na Figura 11. De acordo com Torres et al. (2015), estes vértices estão na margem de separação desde que não exista sobreposição entre as classes.

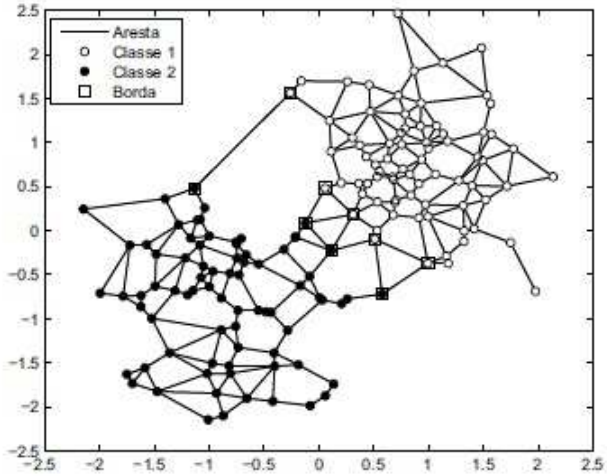

Figura 11 – Grafo de Gabriel com identificação de bordas. Adaptado de (TORRES, 2012)

3.4.2 Eliminação de sobreposição

Em seu trabalho Torres (2016) descreve uma forma de lidar com problemas em que existam ruídos e/ou sobreposição entre as classes. Isto é mostrado na Figura 12. Aupetit e Catz (2005) propôs uma medida de qualidade dos vértices, baseado em seu grau, sendo que o grau de um vértice representa o número de arestas conectadas a ele, essa medida de qualidade é definida por:

$$q(x_i) = \frac{\widehat{\mathcal{A}}(x_i)}{\mathcal{A}(x_i)}, \tag{3.2}$$

sendo que $\mathcal{A}(x_i)$ é o grau do vértice x_i e $\widehat{\mathcal{A}}(x_i)$ representa o grau de x_i menos os vértices de classes distintas de x_i.

Com o valor de $q(x_i)$, o vértice x_i pode ser classificado de 3 formas distintas:

- $q(x_i) = 0$, representa um ruído, uma vez que todos os vértices que compartilham uma aresta com x_i são de classes diferentes de x_i.

- $q(x_i) = 1$, todos os vértices que compartilham uma aresta com x_i são da mesma classe de x_i.

- $0 < q(x_i) < 1$, neste caso, x_i compartilha aresta com vértices de ambas as classes. Portanto x_i é um candidato ao conjunto de Vetores Geométricos (VG).

Para determinar se os pontos que possuem $0 < q(.) < 1$ são considerados ruídos, Torres (2016) propõe a definição de um limiar a partir da medida de qualidade $q(.)$ de todos os dados separados em classes. Torres (2016) descreve o método nas etapas que se seguem:

- Para todo $x_i \in G$, calcule $q(x_i)$.

- Agrupar $q(x_i)$ por classe, tal que Q^+ e Q^- representam a medida de qualidade para os padrões com os rótulos +1 e -1, respectivamente.

- Calcular o valor do limiar t^+ e t^- de cada classe como a média da medida de qualidade pertencendo à Q^+ e Q^-, sendo que

$$t^+ = \frac{\sum q(x_i)}{|Q^+|}, \quad para \quad q(x_i) \in Q^+; \tag{3.3}$$

$$t^- = \frac{\sum q(x_i)}{|Q^-|}, \quad para \quad q(x_i) \in Q^-. \tag{3.4}$$

- Remover de G todos os vértices cuja $q(x_i)$ é menor que t^+ e t^-.

Na Figura 12 é possível observar dados que exemplificam os três casos descritos. o vértice x_1 possui valor $q(x_1) = 0$, pois suas arestas ligam vértices de outras classes, portanto x_1 representa um ruído. Já o vértice x_2 possui valor $q(x_2) = 1$ indicando que suas arestas ligam vértices de mesma classe, dessa forma, x_2 não fornece informação sobre a borda e também não é considerado um ruído. Já o vértice x_3 possui valor $q(x_3) = 2/3$ indicando que o vértice x_3 possui 3 arestas, sendo que duas delas ligam vértices de mesma classe e uma liga vértice de outra classe, assim o vértice x_3 é um candidato a Vetor Geométrico (VG).

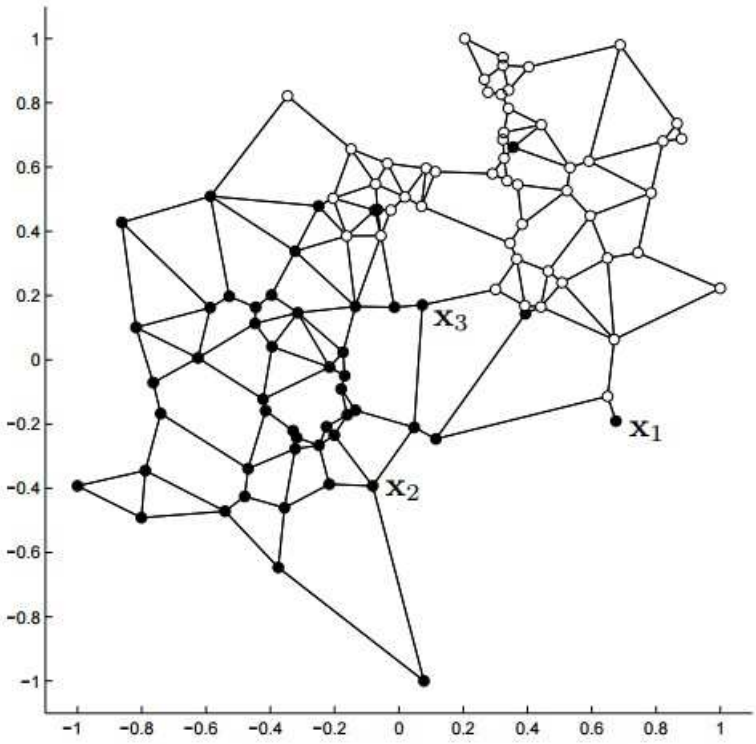

Figura 12 – Dados com sobreposição. Adaptado de (TORRES, 2016)

3.4.3 Classificador Geométrico obtido por combinação de funções lineares

O classificador proposto por Torres et al. (2015), denominada *Chip-clas*, é baseado na combinação de classificadores de margem máxima definidos por todos pares de Vetores Geométricos. Nesta abordagem, a combinação é ponderada por uma função de nível de compatibilidade baseada na distância Euclidiana entre uma amostra a ser classificada e cada hiperplano.

Seja H_i um hiperplano local gerado entre duas amostras $(\mathbf{v}_1, \mathbf{v}_2)$ onde o rótulo de $(\mathbf{v}_1) = -1$ e $(\mathbf{v}_2) = +1$, como ilustrado na Figura 13. O hiperplano passa exatamente pelo ponto mediano (ponto médio) entre $\mathbf{v}_1$ e $\mathbf{v}_2$. Para cada hiperplano H_i existira um ponto médio $\mathcal{PM}_i$ (TORRES et al., 2015).

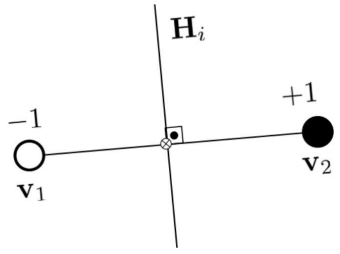

Figura 13 – Definição do hiperplano local entre o par $(\mathbf{v}_1, \mathbf{v}_2)$ do grafo de Gabriel

A expressão que define H_i é dada por (BENNETT; BREDENSTEINER, 2000), e descrita

como

$$H_i(\mathbf{x}, \mathbf{w}_i, b_i) = \mathbf{x}^T \mathbf{w}_i - b_i \tag{3.5}$$

sendo $\mathbf{w}_i = \mathbf{v}_1 - \mathbf{v}_2$ o vetor normal do hiperplano onde sua orientação é descrita através das seguintes inequações

$$\mathbf{x}^T \mathbf{w}_i - b_i \geq 0,$$
$$\mathbf{x}^T \mathbf{w}_i - b_i < 0$$

onde o *bias* b_i é calculado como

$$b_i = \left(\frac{\mathbf{v}_1 + \mathbf{v}_2}{2} \right) \mathbf{w}_i^T. \tag{3.6}$$

3.4.3.1 Classificação

Dado um conjunto de m hiperplanos locais $\{H_1, H_2, \ldots, H_m\}$, a classificação de uma amostra arbitrária $\mathbf{x}_j$ é dada pelos seguintes passos

1. Calcular a saída $\tilde{y}_{i,j}$ para todos H_i

$$\tilde{y}_{i,j} = H_i(\mathbf{x}_j, \mathbf{w}_i, b_i), \quad \forall i = 1, \ldots, m. \tag{3.7}$$

   e distância

$$d_{i,j} = \delta^2 \left(\mathbf{x}_j, \mathcal{PM}_i \right) \ \forall i = 1, \ldots, m \tag{3.8}$$

   onde $d_{i,j}$ é a distância euclidiana entre $\mathbf{x}_j$ e todos os pontos médios.

2. Obter o nível de compatibilidade da amostra $\mathbf{x}_j$ para o i-ésimo hiperplano H_i

$$D_{i,j} = \frac{1}{d_{i,j} \sum_{i=1}^{m} d_{i,j}}, \quad \forall \quad i =, ..., m. \tag{3.9}$$

   e

$$i* = \arg\min \ D_{i,j} \quad i = 1, ..., m. \tag{3.10}$$

   sendo $\arg\min$ o menor valor em um conjunto de dados.

3. Por fim, a função de decisão $f(\mathbf{x}_j)$ pode ser estimada apenas pelo hiperplano mais próximo.

$$f(\mathbf{x}_j) = \begin{cases} +1 & \text{se } D_{i*,j} \ sgn\left(\tilde{y}_{i*,j}\right) \geq 0 \\ -1 & \text{caso contrário} \end{cases} \tag{3.11}$$

   sendo *sgn* a função que retorna o sinal do rótulo.

A Figura 14 exemplifica as etapas utilizadas pelo classificador. Na Figura 14(a) temos um problema com duas classes linearmente separáveis. A Figura 14(b) mostra o Grafo de Gabriel construído. Na 14(c) temos somente as arestas do Grafo de Gabriel que ligam os Vetores Geométricos (VGs). A Figura 14(d) indica, também, os pontos médio entre os VGs. A Figura 14(e) tem-se todos hiperplanos gerados pelos pares de VGs e por fim, tem-se na Figura 14(f) o hiperplano final. Nota-se que o hiperplano, além de separar as duas classes, também maximiza as margens.

3.5 Considerações

A utilização do Grafo de Gabriel, e consequente obtenção dos Vetores Geométricos, fornece uma característica importante para implementação em algoritmos incrementais, uma vez que os Vetores Geométricos carregam informações da borda de separação entre as classes, ou seja, são os dados relevantes para a obtenção de um classificador de margem larga, podendo os demais serem descartados sem perder informação da margem de separação. Outra característica da técnica apresentada está no fato da não utilização de parâmetros fornecidos por um especialista. Em um algoritmo incremental que necessita recorrer ao classificador diversas vezes a não utilização de parâmetros é um ganho interessante.

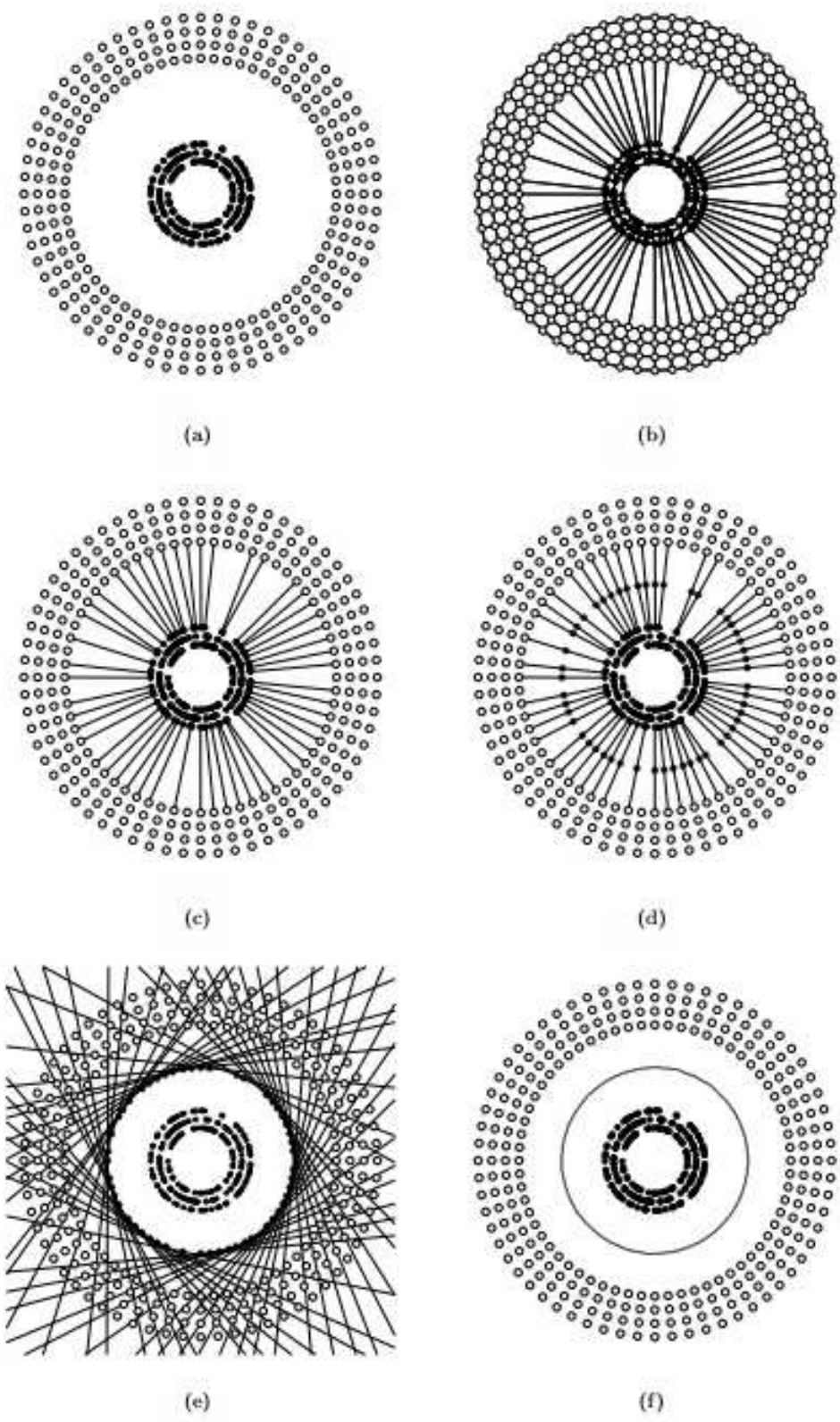

Figura 14 – Etapas classificador Geométrico obtido por combinações de funções lineares. Adaptado de (TORRES, 2016)

4 Metodologias

O Trabalho visa propor técnicas incrementais utilizando a estrutura do Grafo de Gabriel. Dessa forma, serão apresentadas algumas metodologias, divididas em dois grupos. O primeiro grupo baseia-se unicamente nas informações das bordas entre as classes, sendo que, esta informação será extraída após a obtenção do G_G. O segundo grupo baseia-se, além da informação das bordas entre as classes, em informações de dados muitos distantes do centroide da distribuição dos dados. As seções seguintes serão destinadas a descrever as metodologias.

As técnicas incrementais desenvolvidas atuam no armazenamento dos dados (memória), onde o sistema deve ser capaz de descartar dados considerados irrelevantes. A Figura 15 mostra o esquema genérico das técnicas desenvolvidas. Neste esquema notamos que o processo de esquecimento ocorre na memória. Os dados armazenados em memória passam por processos de filtragem e seleção de dados relevantes para o conjunto de treinamento. Com isso, os dados não selecionados são descartados de forma definitiva. De acordo com a literatura, essa técnica de atuação na memória faz com que os algoritmos desenvolvidos sejam denominados Algoritmos Incrementais com memória parcial.

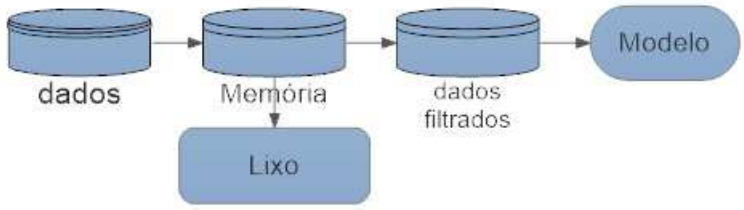

Figura 15 – Esquema genérico das técnicas incrementais implementadas.

Antes de iniciarmos as descrições das metodologias implementadas, definiremos algumas siglas importantes para o entendimento das próximas seções.

- JT - Janela Temporária - recebe o fluxo de dados de entrada e os armazena temporariamente. Assim que os dados presentes em JT passam para outra etapa do processo incremental, esta janela (JT) excluíra todos os dados. A menos que ocorra uma realimentação de dados, a JT sempre estará vazia para receber novos dados.

- Filtro GG - Filtro Grafo de Gabriel - sua função é selecionar os dados que estão nas bordas das classes, através da estrutura do *GG* e da determinação dos vetores geométricos, como descrito em capítulos anteriores. Os vetores geométricos são selecionados para a etapa seguinte, já os demais dados são descartados.

- Filtro Ruído - A filtragem de ruído ocorre como descrito na Seção 3.4.2, ou seja, com o *GG* construído define-se a medida de qualidade ($q(x_i)$) dos dados e, então os dados que possuírem valor de $q(x_i)$ abaixo do limiar são considerados como ruído, sendo, portanto, descartados.

- JG - Janela Geométrica - sua função é armazenar os dados filtrados de forma contínua, ou seja JG estoca os dados e, diferentemente de JT, não realiza o esvaziamento com a chegada de novos dados. Após a filtragem dos dados presentes em JG, é realizado a atualização e JG passa a comportar somente os dados filtrados. A função de JG é armazenar informação de todas as janelas de dados que passaram pelo Algoritmo

4.1 Algoritmo *Batch*

O Algoritmo *Batch*, mostrado na Figura 16, descarta o modelo obsoleto, sempre que um novo conjunto de dados surge, para gerar um novo modelo. Essa estratégia é bem simples, mas descumpre umas das exigências para que o algoritmo seja considerado incremental, pois não armazena conhecimentos anteriores.

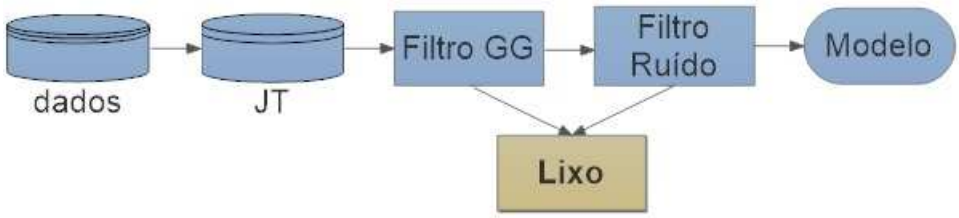

Figura 16 – Algoritmo Batch.

No Algoritmo *Batch* os dados chegam e são armazenados em uma Janela Temporária (JT), os dados armazenados em JT passam pelo Filtro GG, o qual seleciona somente as arestas de suporte e descarta os demais dados. A segunda parte consiste em eliminar ruídos dos dados selecionados pelo Filtro GG, após, o classificador é obtido utilizando o método chip-clas (TORRES et al., 2015). Vale salientar que JT estará vazia, pois tem a função de armazenamento temporário. Após a chegada de novos dados o processo se repete eliminando o classificador anterior.

4.2 Algoritmo 1 (IncV0)

O IncV0 surgiu com o intuito de acrescentar o armazenamento de informações anteriores e cumprir os requisitos para ser considerado um algoritmo incremental. O IncV0,

mostrado na Figura 17, difere do algoritmo Batch pois possui uma realimentação em JT, ou seja, após as filtragens, os dados selecionados são armazenados novamente em JT.

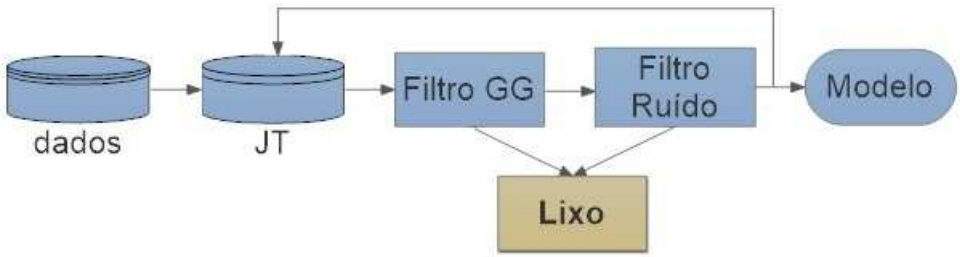

Figura 17 – Algoritmo 1 - IncV0.

No IncV0 os dados que chegam, de forma temporal, são armazenado em JT, respeitando a capacidade de armazenamento, e passam pelo Filtro GG e pelo Filtro Ruído. Os dados selecionados são, então, utilizados para gerar o classificador, mas são, também, realimentados e armazenados novamente em JT. Os novos dados que chegarem serão armazenados em JT, juntamente com os dados antigos, que foram realimentados e armazenados em JT. Dessa maneira, JT possuirá informações novas e antigas. O processo de filtragem e realimentação é repetido com os novos dados de JT. Essa estratégia garante o armazenamento de informações relevantes de todas as janelas que passaram pelo algoritmo.

Essa metodologia possui o problema de utilizar JT para juntar os dados novos e antigos. Isso pode gerar um atraso no tratamento dos dados, uma vez que, o número de dados novos, juntamente com os dados antigos, possa ultrapassar a capacidade de JT, fazendo com que uma certa quantidade de dados novos tenha que "esperar"a liberação de espaço em JT.

4.3 Algoritmo 2 (IncV1)

O Algoritmo IncV1 tem como principal diferença do IncV0 a existência de duas janelas de armazenamento, com o intuito de eliminar o problema de congestionamento de JT, presente no Algoritmo IncV0. O IncV1 difere do IncV0 pois, além de JT, possui a Janela Geométrica (JG) que tem a função de armazenar dados filtrados de JT, vide Figura 18. JG não realiza o descarte dos demais dados com a chegada de novos dados, como acontece com JT, sendo assim, JG tem a função de armazenamento permanente. Mesmo tendo uma característica de armazenamento permanente, JG sofre atualizações sempre que seus dados passam pela filtragem. Dessa maneira, os dados de JG, selecionados pela filtragem, são novamente adicionados a JG e os demais são descartados.

No IncV1 os dados que chegam, são armazenado em JT e passam pelo Filtro GG e pelo Filtro Ruído. Os dados selecionados são, então, armazenados em JG. Na primeira

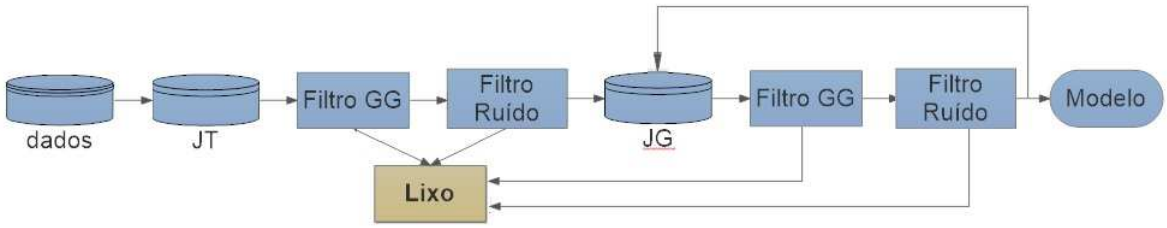

Figura 18 – Algoritmo 2 - IncV1.

iteração, JG possuíra os dados de JT filtrados na primeira iteração, já na segunda iteração, JG possuíra os dados de JT filtrados na primeira e segunda iterações. Os dados de JG são novamente filtrados e o modelo gerado. Os dados filtrados retornam para JG, que descarta os demais, como ilustrado na Figura 18. O processo prossegue com o armazenamento, em JG, dos dados filtrados de JT, juntamente com os demais dados presentes da iteração anterior.

4.4 Algoritmo 3 (IncV2)

O Algoritmo IncV2 difere do IncV1 na filtragem dos dados de JT, uma vez que no IncV2 ocorre somente a filtragem com o Filtro GG, no IncV1 ocorre a filtragem com o Filtro GG e o Filtro Ruído. Esta diferença é motivada no fato da filtragem de ruído confundir dados importante com ruídos. Como JT possui uma parcela de dados relativamente reduzida, se comparada com o total de dados, que muitas vezes é infinito, pois os dados chegam de forma temporal e não possui fim, a detecção de ruídos se torna uma tarefa um pouco complicada.

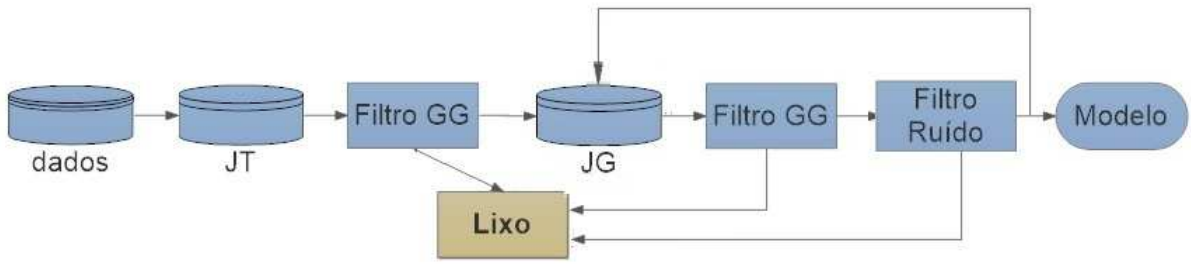

Figura 19 – Algoritmo 3 - IncV2.

No IncV2 os dados que chegam, são armazenados em JT e passam pelo Filtro GG. Os dados selecionados são, então, armazenados em JG. Na primeira iteração, JG possuíra os dados de JT filtrados na primeira iteração. Já na segunda iteração, JG possuíra os dados de JT filtrados na primeira e segunda iterações. Os dados de JG passam, então, pelos Filtro GG e Filtro Ruído e o modelo é gerado. Os dados filtrados retornam para JG, que

descarta os demais, Figura 19. O processo prossegue com o armazenamento, em JG, dos dados filtrados de JT juntamente com os demais dados presentes da iteração anterior.

A Figura 20 mostra um exemplo de como a filtragem e seleção se comporta, neste exemplo os dados foram divididos em dez janelas. Os dados em azul e vermelho são os dados selecionados nas janelas JT. Os dados circulados são os dados selecionados em JG, neste caso o modelo foi obtido somente no fim do processo, depois da filtragem das dez janelas JT. Os dados em verde são os dados descartados durante as filtragens de JT, nota-se que uma gama de dados foram descartados sem prejudicar a manutenção da margem de separação.

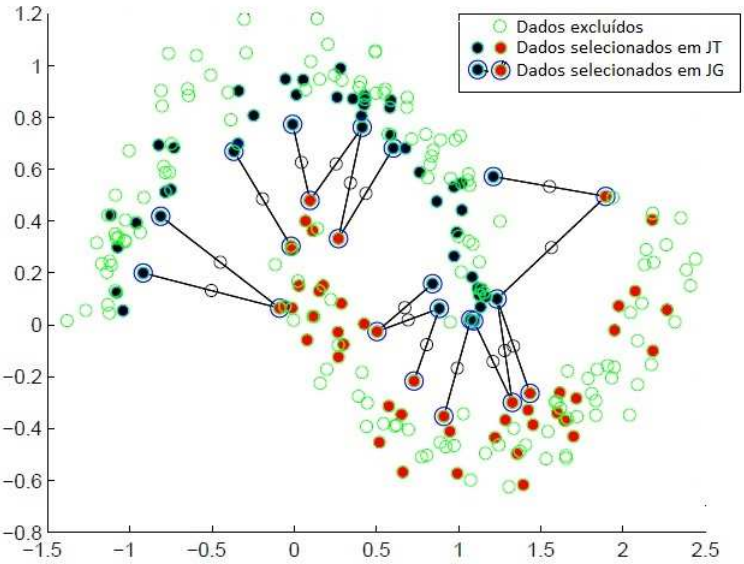

Figura 20 – Seleção de dados pelo algoritmo IncV2.

4.5 Algoritmo 4 (IncV3)

O Algoritmo IncV3 possui duas etapas de seleção de dados considerados relevantes. A primeira consiste nos processos de filtragem descritos acima, Filtro GG e Filtro Ruído. A segunda, chamada de Quarentena, consiste em selecionar dados muito distantes da distribuição dos dados de uma dada JT.

A ideia de utilizar a quarentena surge de problemas onde a distribuição dos dados pode sofrer mudanças gradativas. Nessas mudanças gradativas os Filtros GG e Ruído poderiam eliminar esses novos dados. A Figura 21 exemplifica esse problema.

Na Figura 21(a) pode ser visto os dados selecionados para JT e os dados futuros, que ainda não são conhecidos pelo sistema. A Figura 21(b) mostra os dados selecionados pelos filtros, como pode ser observado, o filtro descartou informações importantes para a geração do modelo. A Figura 21(c) mostra o classificador encontrado ao final da chegada de todos os dados. Nota-se que a eliminação da informação comprometeu o desempenho

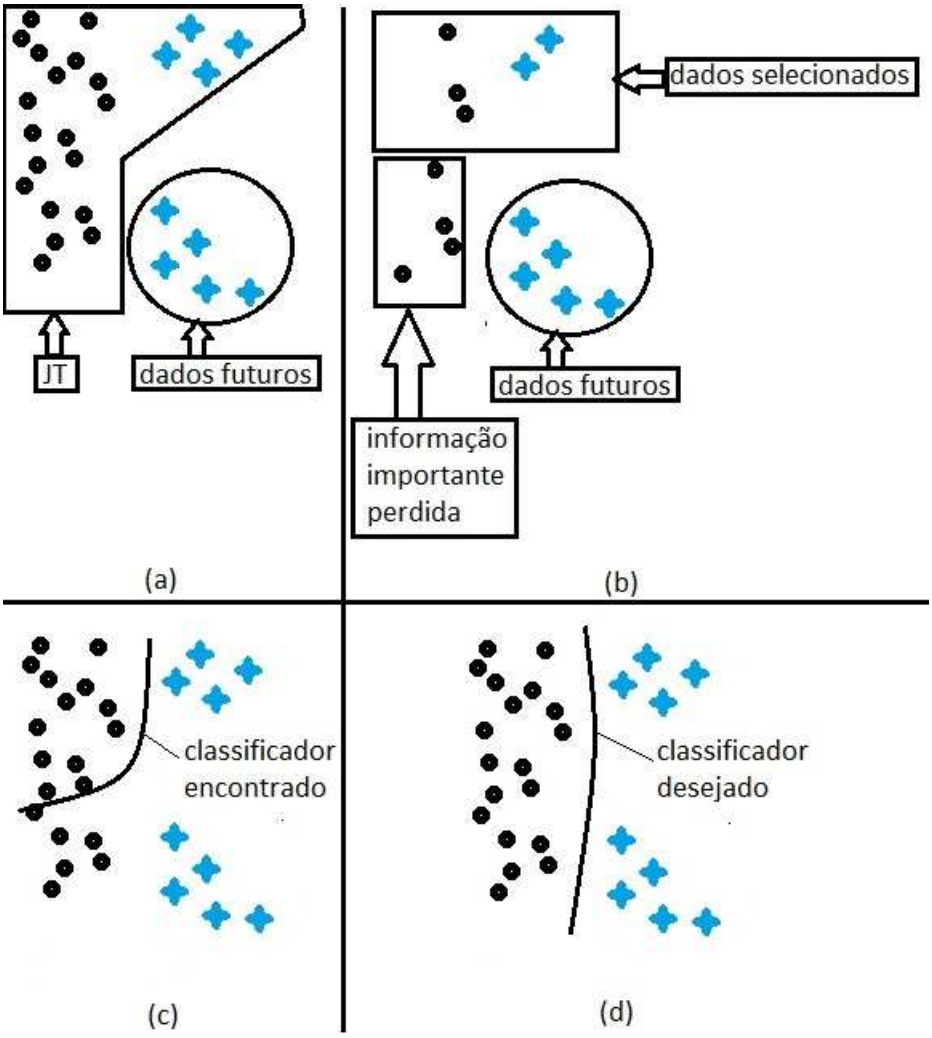

Figura 21 – Problema da mudança de distribuição.

do classificador, uma vez que o classificador desejado é como se mostra na Figura 21(d). Dessa forma, faz-se necessário desenvolver um mecanismo que trate esse problema de eliminação de informações relevantes.

O IncV3 possui um mecanismo que tenta suprir essa deficiência em armazenar informações de uma possível mudança na distribuição dos dados. Esse mecanismo foi denominado como Quarentena. Neste processo, são determinados uma quantidade pré definida de dados mais distantes da média da distribuição, ou seja, encontra-se a média espacial dos dados presentes em JT e seleciona, os dados de JT mais distantes dessa média, esses dados são armazenados em uma janela chamada Quarentena, onde ficarão por lá enquanto sua idade não atingir a idade máxima. A cada iteração, ou seja, a chegada de novos dados em JT, a idade dos dados é incrementada em uma unidade. Os dados presentes na Quarentena são repassados para JG, juntamente com os dados filtrados de JT. Neste procedimento é como se os dados da Quarentena estivessem "aguardando"novos dados que pudessem se unir e formar novos Vetores Geométricos. A Figura 22 mostra o funcionamento do Algoritmo IncV3.

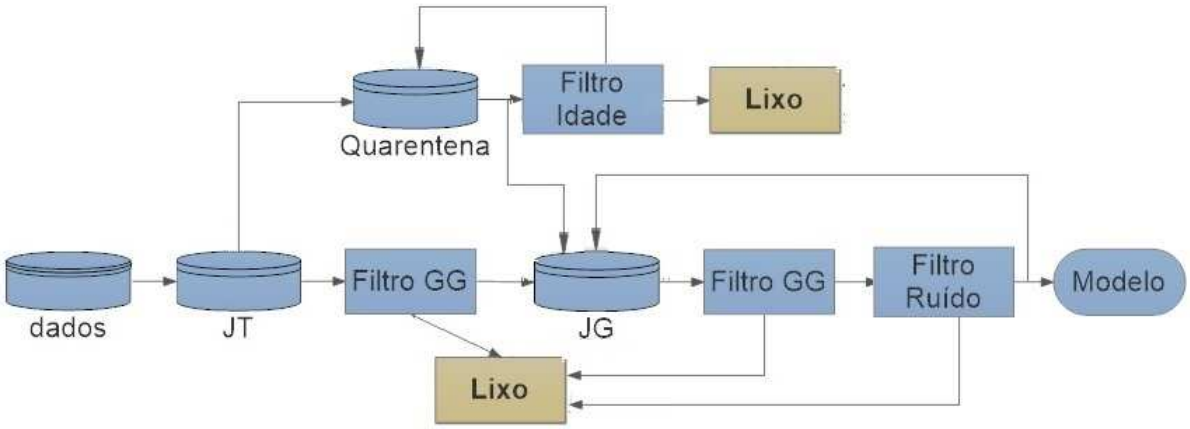

Figura 22 – Algoritmo 4 - IncV3.

4.6 Discussões do capítulo

Dados os quatro algoritmos incrementais, além do *batch*, foi realizado um comparativo entre as metodologias, apresentando um resumo das principais características de cada estratégia, pontos positivos e pontos negativos. A Tabela 1 mostra estas características.

Tabela 1 – Características principais das técnicas incrementais.

Algoritmo	Caracterısticas Principais	Principal ponto positivo	Principal ponto negativo
Batch	Renova todo conjunto com a presença de novos dados;	Janelas sempre com o mesmo tamanho;	Não cumpre pré-requisitosdos algoritmos incrementais;
IncV0	Realimentação,em JT, dos dados filtrados;	Implementação simples, cumpre os pré-requisitos dos algoritmos incrementais;	Pode gerar congestionamento de dados de entrada, em JT;
IncV1	Dados filtrados armazenados em outra janela (JG);	Evita congestionamento dos dados de entrada;	Não possui mecanismo explícito de detecção de mudança na distribuição dos dados;
IncV2	Filtro Ruído utilizado somente em JG;	Evita perda de informação importante causada pela filtragem de ruído em JT;	Não possui mecanismo explícito de detecção de mudança na distribuição dos dados;
IncV3	Duas técnicas distintas de seleção de dados.	Possui mecanismo explícito de detecção de mudança na distribuição dos dados.	Necessário definir parâmetros, a priori.

5 Resultados e Discussões

Com a implementação das metodologias descritas anteriormente, foram realizados alguns testes, afim de validar as metodologias e também comparar as técnicas. Os testes foram realizados durante a evolução do trabalho, inicialmente validou-se e testou o IncV0, em seguida o IncV1 e IncV2 e finalmente o IncV3. Testes comparativos foram realizados com o intuito de comparar as metodologias implementadas com as técnicas presentes na literatura.

5.1 Validação do algoritmo IncV0

O primeiro teste realizado consiste na validação da metodologia incremental através da comparação com a metodologia tradicional. A metodologia tradicional considera uma situação hipotética, onde todos os dados estão disponíveis e é possível utilizá-los, de uma só vez, em um algoritmo tradicional e gerar o modelo preditor. Vale salientar que esta situação não ocorre em um cenário *data stream* e *big data*. Para analisar o desempenho do algoritmo IncV0, serão utilizados três bases de dados *benchmark* para avaliação de classificadores (HETTICH; BLAKE; MERZ, 1998), *Sonar*, *Parkinson* e *ILPD*. Os dados foram distribuídos em conjuntos de treinamento e teste e avaliados utilizando duas metodologias. Uma delas consiste em treinar um modelo utilizando todos os dados de treinamento e avaliar o modelo com os dados de teste, ou seja, a metodologia tradicional. A segunda metodologia utilizada foi o IncV0, da mesma forma, o conjunto foi avaliado através dos dados de teste.

Para cada base de dados foi realizado um teste de hipótese estatística t pareado (MONTGOMERY, 2006) a fim de verificar a equivalência dos métodos incremental e tradicional, para isso foram realizadas 30 execuções de cada algoritmo para cada base de dados. O teste consiste em analisar se as médias dos modelos são equivalentes, para isso utilizou-se um nível de significância de 0.05 e uma potência de 80%. Abaixo temos o teste de hipótese realizado.

$$\begin{cases} H_0 : \mu_1 - \mu_2 = 0 \\ H_1 : \mu_1 - \mu_2 \neq 0 \end{cases}$$

O teste de hipóteses foi realizado de duas formas, a primeira consiste em avaliar o desempenho dos algoritmos individualmente para cada base, de forma a verificar a influência do tipo de problema abordado no desempenho dos classificadores. A segunda análise foi feita através de um teste estatístico com dados de todas as bases, a fim de verificar o desempenho mais geral.

5.1.1 Sonar

Esta base de dados foi utilizada por Gorman e Sejnowski (1988) no estudo sobre classificação de dados de Sonar na diferenciação de rochas cilíndricas e um metal cilíndrico. A base possui 208 amostras, cada um com um conjunto de 60 características e uma característica indicando a qual classe o dado pertence. A Figura 23 mostra um boxplot do desempenho das duas metodologias para um total de 30 execuções para cada uma delas.

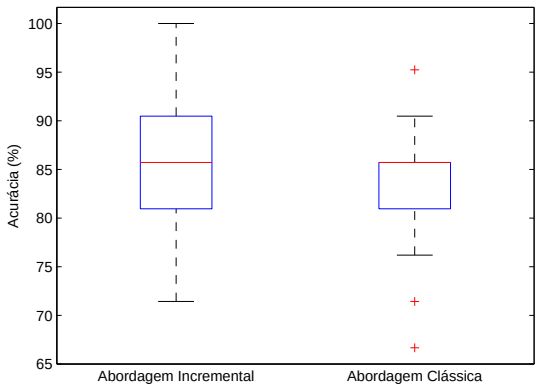

Figura 23 – Desempenho dos classificadores para a base Sonar.

A Figura 24 mostra a comparação das médias das metodologias com um intervalo de confiança de 95%. Nela, é possível verificar a equivalência das médias, uma vez que a diferença zero está dentro do intervalo de confiança, dessa forma não podemos rejeitar a hipótese de igualdade das médias

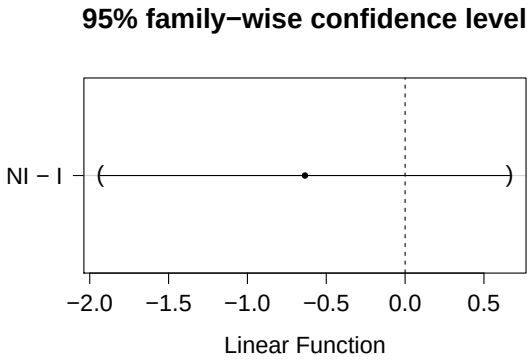

Figura 24 – Intervalo de Confiança para comparação de desempenho médio. NI - Não Incremental, I - Incremental.

5.1.2 Parkinson

Esta base de dados foi utilizada por Little et al. (2007) no estudo de distúrbios de voz. A base possui 195 amostras, cada um com um conjunto de 23 características e uma

característica indicando a qual classe o dado pertence. A Figura 25 mostra um *boxplot* do desempenho das duas metodologias para um total de 30 execuções para cada uma delas.

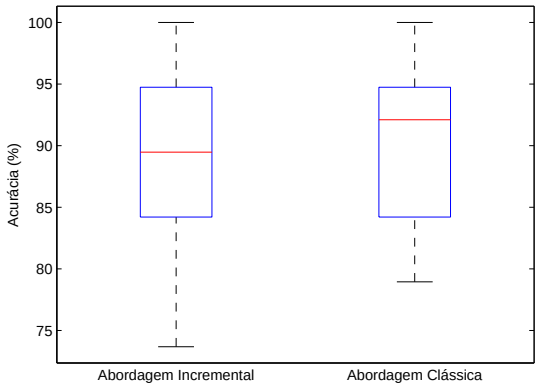

Figura 25 – Desempenho dos classificadores para a base Parkinson.

A Figura 26 mostra a comparação das médias das metodologias com um intervalo de confiança de 95%. Nela, é possível verificar a equivalência das médias, uma vez que a diferença zero está dentro do intervalo de confiança, dessa forma não podemos rejeitar a hipótese de igualdade das médias

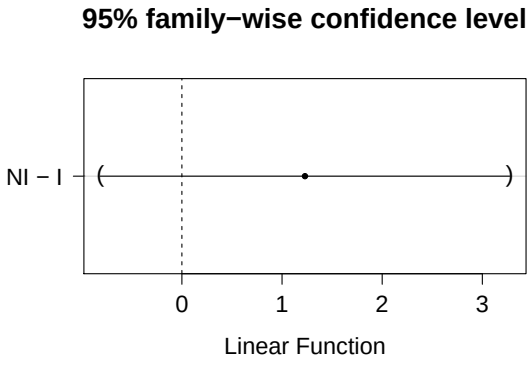

Figura 26 – Intervalo de Confiança para comparação de desempenho médio. NI - Não Incremental, I - Incremental.

5.1.3 ILPD

Esta base possui 583 amostras, cada um com um conjunto de 10 características e uma característica indicando a qual classe o dado pertence (LICHMAN, 2013). A Figura 27 mostra um boxplot do desempenho das duas metodologias para um total de 30 execuções para cada uma delas.

A Figura 28 mostra a comparação das médias das metodologias com um intervalo de confiança de 95%. Nela, é possível verificar a equivalência das médias, uma vez que a

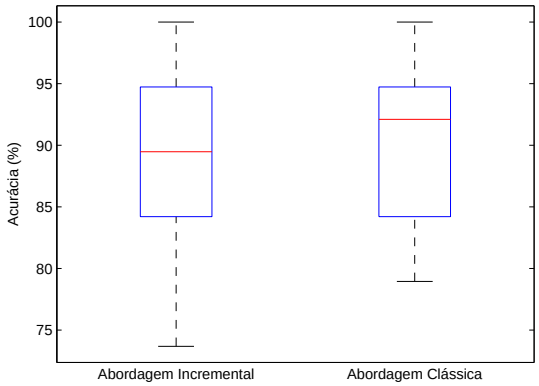

Figura 27 – Desempenho dos classificadores para a base ILPD.

diferença zero está dentro do intervalo de confiança, dessa forma não podemos rejeitar a hipótese de igualdade das médias

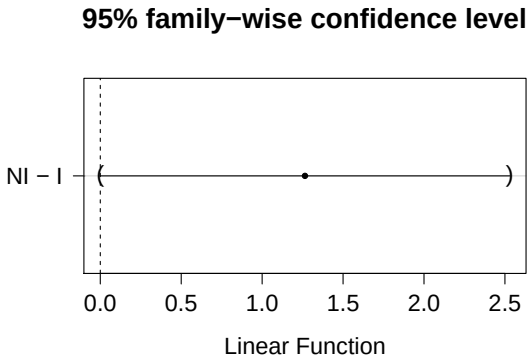

Figura 28 – Intervalo de Confiança para comparação de desempenho médio. NI - Não Incremental, I - Incremental.

A Tabela 2 mostra também a comparação de desempenho, medido através da acurácia, das metodologias clássica e da proposta incremental através de um teste t pareado com 95% de nível de confiança e 80% de potência. Com os testes realizados, não podemos rejeitar a hipótese nula de igualdade de desempenho médio das metodologias.

Tabela 2 – Comparação de Desempenho entre a Metodologia Clássica e a Metodologia Incremental proposta

	Intervalo de Confiança		
Base de Dados	Lim. Inferior	Estimativa	Lim. Superior
Sonar	-1,9332000	-0,6350000	0,6632000
Parkinson	-0,8149000	1,2290000	3,2729000
ILPD	-0,0000147	1,2650000	2,5290000

Para os testes realizados, a execução de 30 rodadas de cada abordagem foi suficiente para garantir a potência requerida de 80% para os testes estatísticos, tornando válida a

não rejeição da hipótese nula de que as abordagens são equivalentes.

5.2 Validação dos algoritmos IncV1 e IncV2

Um segundo teste realizado, consiste em analisar a desempenho do IncV1 se comparado a outras técnicas incrementais. Para isso, serão utilizados 5 algoritmos, o algoritmo Batch, IncV0, IncV1, IncV2, além de uma outra metodologia, a qual consiste em substituir o Filtro GG e o Filtro Ruído do IncV1 pela seleção de vetores de suporte fornecidos pelo classificador SVM, ou seja os dados selecionados pelo filtro são os vetores geométricos das janelas JT e JG. Vale salientar que o treinamento da SVM, além de selecionar os vetores de suporte elimina possíveis ruídos, através do mapeamento dos dados (BURGES, 1998). Os parâmetros da SVM utilizada foram ajustados por validação cruzada sempre que um novo conjunto de dados deveria ser avaliado.

A métrica utilizada para avaliar os dados foi a área abaixo da curva ROC (AUC). A curva ROC é baseada na taxa de verdadeiros positivos *tpr* e na taxa de falsos positivos *fpr*. De acordo com (FAWCETT, 2006) tpr é dita a sensibilidade e fpr 1-especificidade do sistema. Uma maneira de extrair um valor de desempenho é calculando a área sob a curva Roc (AUC) (PRATI; BATISTA; MONARD, 2010). A AUC varia de 0 a 1, sendo que o melhor desempenho é AUC igual a 1.

5.2.1 Métrica de Avaliação

Em (GAMA et al., 2014) é apresentada uma técnica para avaliação de algoritmos incrementais conhecida como *prequential*. De acordo com Rossi (2014) o método *prequential* utiliza um conjunto de dados, desconhecido pelo modelo, para avaliar o preditor. Somente após a avaliação é que os dados podem ser apresentados ao modelo, para que se possa realizar a atualização do mesmo. Ou seja, com a chegada de um novo conjunto de dados, antes que este componha o modelo, é realizado a avaliação do preditor. Dessa forma o modelo é testado por dados ainda desconhecido pelo modelo. O erro então é calculado através de uma soma acumulada de uma função de perda entre a predição e os valores observados. Neste trabalho será aplicado o método *prequential*, onde os dados armazenados em JT serão utilizados para avaliar o modelo em cada instante k_i. Dessa forma, a avaliação representa o cenário real, onde o modelo obtido no instante k_i é utilizado para avaliar dados obtidos já no instante seguinte $k_i + 1$.

Rossi (2014) diz que o método *prequential* apresenta uma avaliação ruim, dado que o conjunto de dados utilizado para teste representa, geralmente, uma porção grande de dados. Além disso, os dados utilizados para teste podem ser provenientes de uma distribuição diferente e comprometer o desempenho do classificador. Sendo assim, ao utilizar essa técnica, não é justo comparar seu desempenho com outras técnicas de avaliação, dado que o *prequential* tem a característica de apresentar baixos resultados de desempenho

5.2.2 Coleta de dados

Para comparar os algoritmos, foram utilizados 11 bases de dados distintas da uci (HETTICH; BLAKE; MERZ, 1998) e mostradas na Tabela 3.

Tabela 3 – Bases de dados utilizadas

Base de dados	#atributos	#dados
Breast Cancer	33	198
Diabetes	08	768
German	24	1000
Heart	13	270
Ionosphere	34	351
Liver Disorder	6	345
Monks1	6	124
Monks2	6	169
Monks3	6	122
Sonar	60	208
Spam Base	57	4601

Para cada base os dados foram divididos em 10 janelas de tamanhos iguais. A cada janela foi utilizado a técnica *prequential* juntamente com a AUC, após a avaliação de toda a base, tira-se a média das avaliações das 10 janelas e tem-se, portanto, um único valor de desempenho. Foram realizadas 30 execuções de cada algoritmo para cada base. Ao final tira-se a média das 30 execuções, tendo, portanto o estimador daquele algoritmo para aquela base. A Tabela 4 mostra os estimadores para as bases e algoritmos considerados.

Tabela 4 – Resultados Coletados para validação de algoritmos

Problema	IncV0	IncV1	IncV2	Batch	SVM
BreastCancer	0.5253	0.5358	0.5923	0.5017	0.6144
Diabetes	0.7554	0.7345	0.7583	0.7579	0.6628
German	0.7249	0.7293	0.7473	0.7248	0.6389
Heart	0.7596	0.7132	0.8181	0.7408	0.6449
Ionosphere	0.8606	0.8347	0.8600	0.8590	0.9721
Liverdisorder	0.6380	0.6166	0.6780	0.6110	0.6098
Monks1	0.4849	0.5366	0.5165	0.5370	0.5039
Monks2	0.4787	0.5026	0.5017	0.5123	0.5144
Monks3	0.4969	0.5165	0.4824	0.5192	0.4813
Sonar	0.6240	0.6065	0.6903	0.6065	0.7217
Spambase	0.7046	0.6742	0.7128	0.6885	0.8084

5.2.3 Análise dos dados

A Figura 29 mostra a disposição dos dados para cada algoritmo e base distinta. Nota-se que os dados se confundem em determinados momentos, tornando difícil verificar se existem diferenças de desempenho entre os algoritmos.

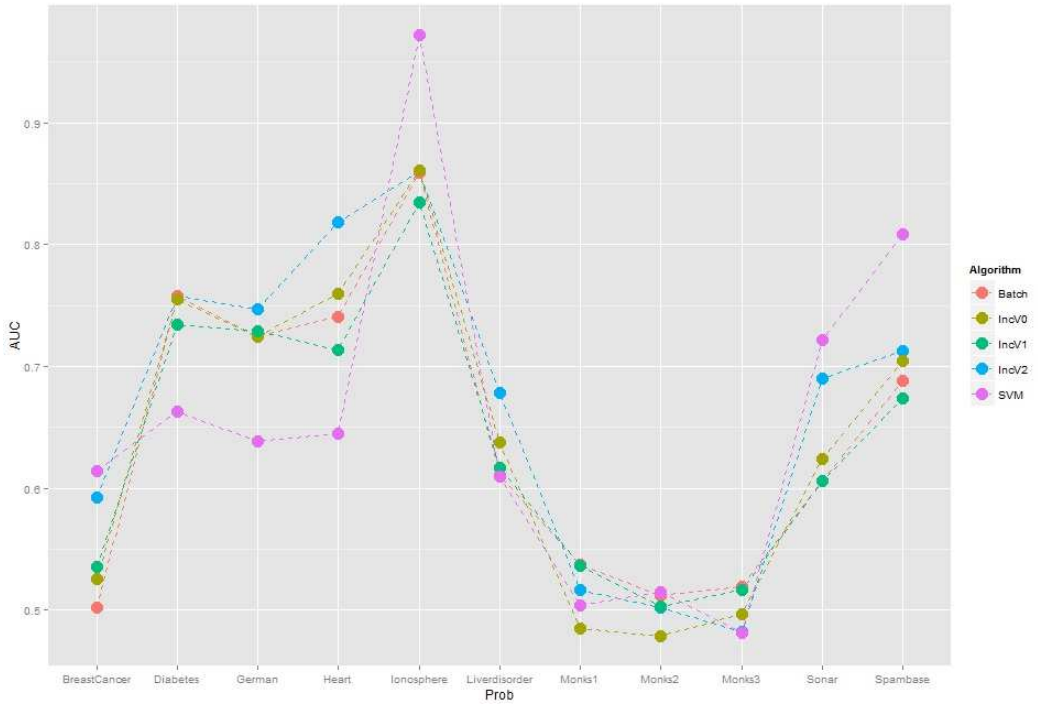

Figura 29 – Distribuição dos resultados dos algoritmos.

Para comparar os algoritmos, foi realizado inicialmente o teste ANOVA (MONTGOMERY, 2006), através do software R (R Development Core Team, 2008). Este teste tem como objetivo verificar se existem diferenças entre os algoritmos comparados, para compor esse teste é necessário realizar o processo de blocagem, que consiste em eliminar os efeitos da diferença entre as bases de dados utilizadas. O teste de hipóteses é mostrado abaixo, onde foi considerado um intervalo de confiança de 95% e uma potência de 80%.

$$\begin{cases} H_0 : \tau_i = 0 \quad \forall \quad i \\ H_1 : \tau_i \neq 0 \end{cases}$$

sendo τ_i o deslocamento, da média global, do algoritmo i. O teste de ANOVA indicou um *p-value* de 0,413. Isso implica em não rejeitar a hipótese nula de igualdade entre os desempenhos dos algoritmos, ou seja, o teste não encontrou diferença entre os 5 algoritmos implementados.

O teste estatístico de comparação múltipla (Dunnett) (MONTGOMERY, 2006) é, normalmente, realizado após o teste de ANOVA rejeitar a hipótese nula, uma vez que o ANOVA, indica se existe, ou não, diferença entre os algoritmos, mas não indica quais são as diferenças, quem é o melhor ou pior algoritmo. Já o teste de comparação múltipla consiste em mostrar estas diferenças. Como o teste de ANOVA não pode rejeitar a hipótese nula, não seria necessário realizar o teste de comparação múltipla. Contudo, as bases de dados utilizadas, são bases relativamente pequenas e podem esconder algumas

diferenças. Além disso, a utilização de 11 bases de dados pode não ter gerado a potência requerida. Dessa forma, foi realizado o teste de comparação múltipla, com o intuito de verificar, pontualmente, se existe diferença entre os algoritmos. O teste Dunnett compara todos algoritmos contra um.

Neste teste foi comparado o algoritmo IncV2 com os demais algoritmos. A Tabela 5 mostra o resultado da comparação. Nesta tabela o campo Estimativa é o valor pontual da diferença de desempenho entre os algoritmos, Inf. é o limite inferior do intervalo de confiança e Sup. é o limite superior do intervalo de confiança. Verifica-se que os intervalos de confiança englobam o valor 0, ou seja, confirmando o que o teste de ANOVA indicou. Porém, analisando o estimador pontual, verifica-se que o IncV2 foi superior em todas as comparações. A Figura 30 traz uma análise visual e um pouco mais clara de visualizar estas análises.

Tabela 5 – Comparação Dunnet entre algoritmos.

Comparação	Estimativa	Inf.	Sup.
Batch-IncV2	-0,03930	-0,10724	0,02864
IncV0-IncV2	-0,04378	-0,11172	0,02416
IncV1-IncV2	-0,04503	-0,11297	0,02291
SVM-IncV2	-0,02797	-0,09591	0,03997

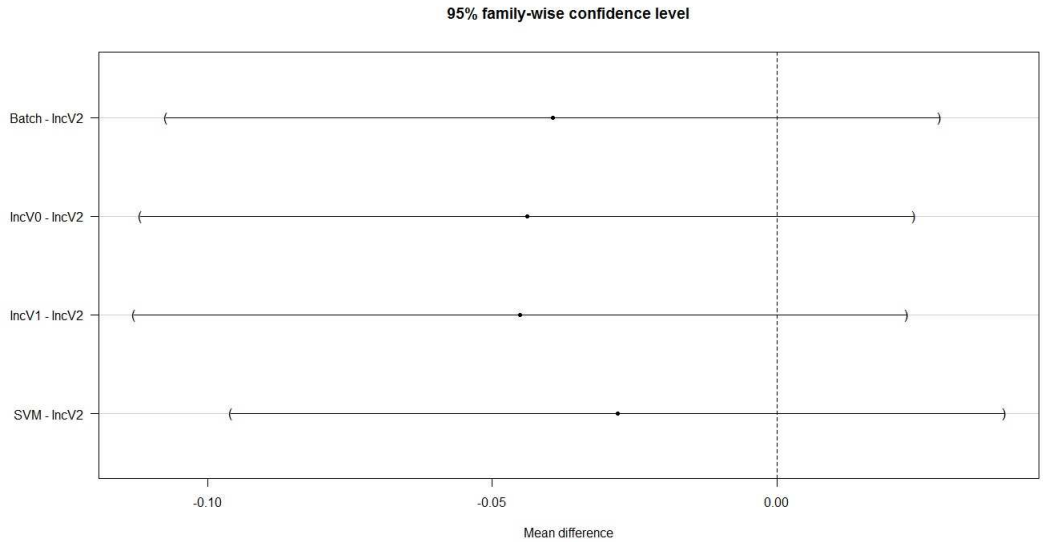

Figura 30 – Comparação Dunnet entre os algoritmos.

5.3 Validação do algoritmo IncV3

Este teste foi realizado a fim de validar o algoritmo IncV3. Esta validação consiste em comparar o IncV3 com o IncV2 que foi validado no teste anterior. Para o IncV3 os

parâmetros utilizados foram: idade igual a 7 iterações e o número de dados selecionados para a quarentena igual a 6, por iteração.

A avaliação ocorrerá como no teste anterior, ou seja, será utilizado a técnica *prequential* juntamente com a AUC para avaliar os métodos.

5.3.1 Coleta de dados

Para comparar os algoritmos, foram utilizados 11 bases de dados distintas da uci (HETTICH; BLAKE; MERZ, 1998) e mostradas na Tabela 6.

Tabela 6 – Bases de dados utilizadas

Base de dados	#atributos	#dados
Breast Cancer	33	198
Diabetes	08	768
German	24	1000
Heart	13	270
Car	6	1728
Euthyroid	24	2000
Satimage	36	6435
Segmentation	19	210
Vehicle	6	846
Vowel	10	990
Yeast	8	1484

Para cada base, os dados foram divididos em 10 janelas de tamanhos iguais. A cada janela foi utilizado a técnica *prequential* juntamente com a AUC em cada janela. Após a avaliação de toda a base, tira-se a média das avaliações das 10 janelas e tem-se, portanto, um único valor de desempenho. Foram realizadas 30 execuções de cada algoritmo para cada base. Ao final tira-se a média das 30 execuções, tendo, portanto o estimador daquele algoritmo para aquela base. A Tabela 7 mostra os estimadores para as bases e algoritmos considerados.

5.3.2 Análise dos dados

A Figura 31 mostra a disposição dos dados para cada algoritmo e base distinta. Nota-se que os dados se confundem em determinados momentos, tornando difícil verificar se existem diferenças de desempenho entre os algoritmos.

Neste caso, em particular, não foi realizado o teste ANOVA, pelo interesse em verificar as diferenças pontuais. Dessa maneira, foi realizado, diretamente, o teste de múltiplas comparações (Dunnet) com o intuito de verificar estas diferenças pontuais. O teste de hipótese foi realizado com os requisitos de 95% de confiança e 80% de potência e pode ser dado como:

Tabela 7 – Resultados Coletados para validação do algoritmo - AUC

Problema	IncV2	IncV3
BreastCancer	0.5966	0.5422
Car	0.7880	0.8796
Diabetes	0.7286	0.7484
Euthyroid	0.7521	0.7884
German	0.6924	0.6959
Heart	0.7521	0.6564
Satimage	0.8231	0.8417
Segmentation	0.9757	0.9658
Vehicle	0.8788	0.7999
Vowel	0.8816	0.7966
Yeast	0.7320	0.8043

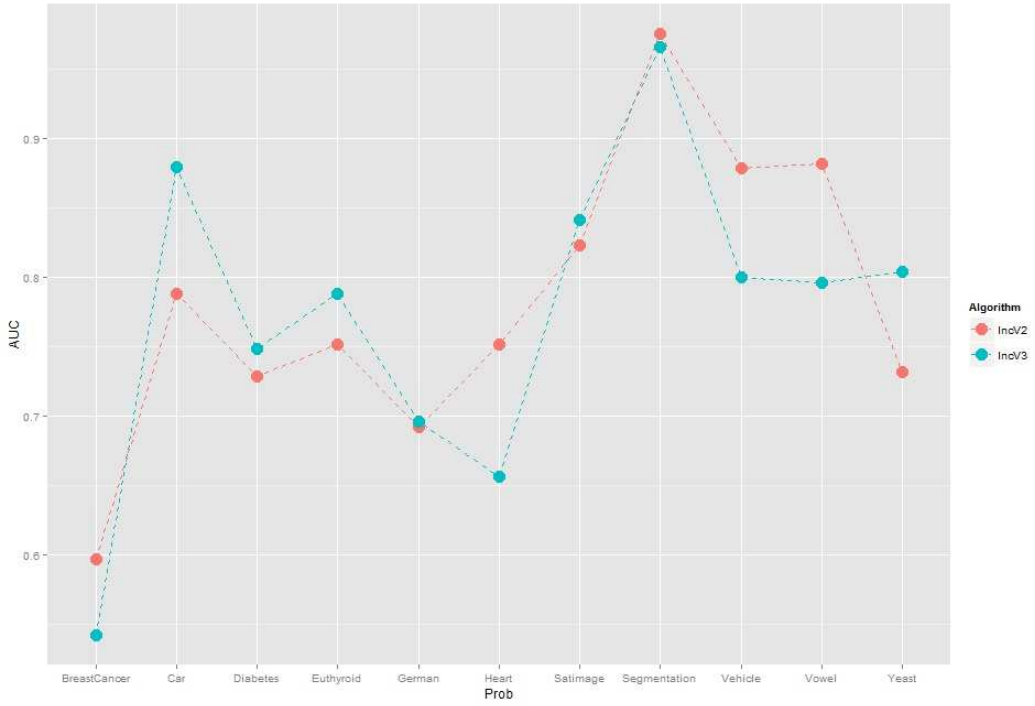

Figura 31 – Distribuição dos resultados dos algoritmos.

$$\begin{cases} H_0 : \tau_i = 0 \quad \forall \quad i \\ H_1 : \tau_i \neq 0 \end{cases}$$

sendo τ_i o deslocamento, da média global, do algoritmo i.

A Tabela 8 mostra o resultado da comparação. Nesta tabela o campo Estimativa é o valor pontual da diferença de desempenho entre os algoritmos, Inf. é o limite inferior do intervalo de confiança e Sup. é o limite superior do intervalo de confiança. Verifica-se que os intervalos de confiança englobam o valor 0, indicando, assim que estatisticamente não existe diferença entre os algoritmos, o estimador pontual também apresentou um valor

pequeno ao indicar que o IncV2 é superior ao IncV3. A Figura 32 traz uma análise visual e um pouco mais clara de visualizar estas análises.

Tabela 8 – Comparação Dunnet entre os algoritmos.

Comparação	Estimativa	Inf.	Sup.
IncV2-IncV3	0,01199	-0,04412	0,06809

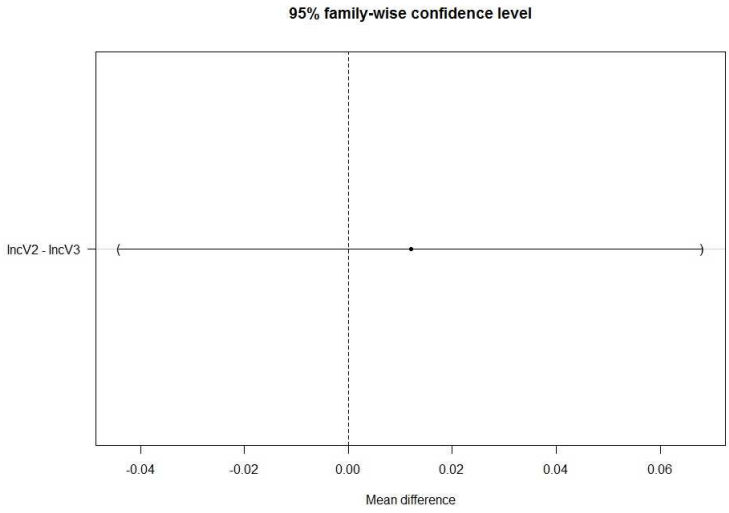

Figura 32 – Comparação Dunnet entre os algoritmos.

5.4 Testes em bases com grande volume de dados

Após os testes de validação, procedeu-se a um teste com bases de dados maiores, em que as bases representassem problemas de *data stream* como por exemplo, os dados de *SPAM* e, também, foram utilizadas bases onde o algoritmo tradicional, que utiliza todos os dados para realizar o treinamento, se tornaria inviável para tratar este tipo de problema.

Foram utilizadas 6 bases da uci (HETTICH; BLAKE; MERZ, 1998), mostradas na Tabela 9, onde pode-se observar o grande volume de dados de algumas delas. Vale salientar que mesmo a base *SPAM* não possuindo um grande volume de dados, se comparado as demais, esta base representa um cenário *data stream*. A construção do Grafo de Gabriel se tornaria inviável, em alguns casos, devido a restrições de tempo, capacidade de armazenamento, construção de matrizes, entre outros entraves.

A comparação dos dados foi realizada utilizando 3 algoritmos, IncV2, IncV3 e o algoritmo proposto por Zheng et al. (2010), denominado, neste trabalho, de ProtOrig. Os parâmetros da SVM utilizada no ProtOrig foram ajustados por validação cruzada.

Tabela 9 – Bases de dados utilizadas.

Base de dados	#atributos	#dados
Spam	57	4601
CodRna	8	271.617
IJCNN1	12	49.990
a1a	123	32.561
SUSY	18	5.000.000
HIGGS	28	1.000.000

5.4.1 Coleta de dados

Cada base os dados foi dividida em janelas de tamanho igual a 400 dados. A cada janela foi utilizado a técnica *prequential* juntamente com a AUC em cada janela. Após a avaliação de toda a base, tira-se a média das avaliações das janelas e tem-se, portanto, um único valor de desempenho. Para a base *SPAM* foram realizadas 30 execuções de cada algoritmo para cada base. Já para a *IJCNN1* e *a1a* foram realizadas 8 execuções. As demais, *CodRna*, *SUSY* e *HIGGS* tiveram 3 execuções cada. Ao final tira-se a média das execuções, tendo, portanto o estimador daquele algoritmo para aquela base. A Tabela 10 mostra os estimadores médios parar as bases e algoritmos considerados. Não foi realizado 30 execuções para todas as bases devido ao grande volume de dados presente na maioria delas. Uma execução dos 3 algoritmos se tornou temporalmente custosa, principalmente a execução do ProtOrig que necessita do ajuste de parâmetros da SVM, por validação cruzada.

Tabela 10 – Estimadores médios AUC para os algoritmos

Problemas	IncV2	IncV3	ProtOrig.
Spam	0.6407146	0.6029592	0.6011292
CodRna	0.637264	0.7406895	0.640276
IJCNN1	0.733452	0.856548	0.858095
a1a	0.543235	0.532321	0.543289
SUSY	0.63213	0.6434	0.6314
HIGGS	0.68976	0.72652	0.709843

A Figura 33 mostra o desempenho dos algoritmos, através de uma análise visual, não é possível detectar se existe diferenças de desempenho dos algoritmos.

Foi realizado o teste de ANOVA para verificar se existe diferença estatística entre os algoritmos. O teste foi realizado com os requisitos de 95% de confiança e 75% de potência, o teste de hipóteses é mostrado abaixo:

$$\begin{cases} H_0 : \tau_i = 0 \quad \forall \quad i \\ H_1 : \tau_i \neq 0 \end{cases}$$

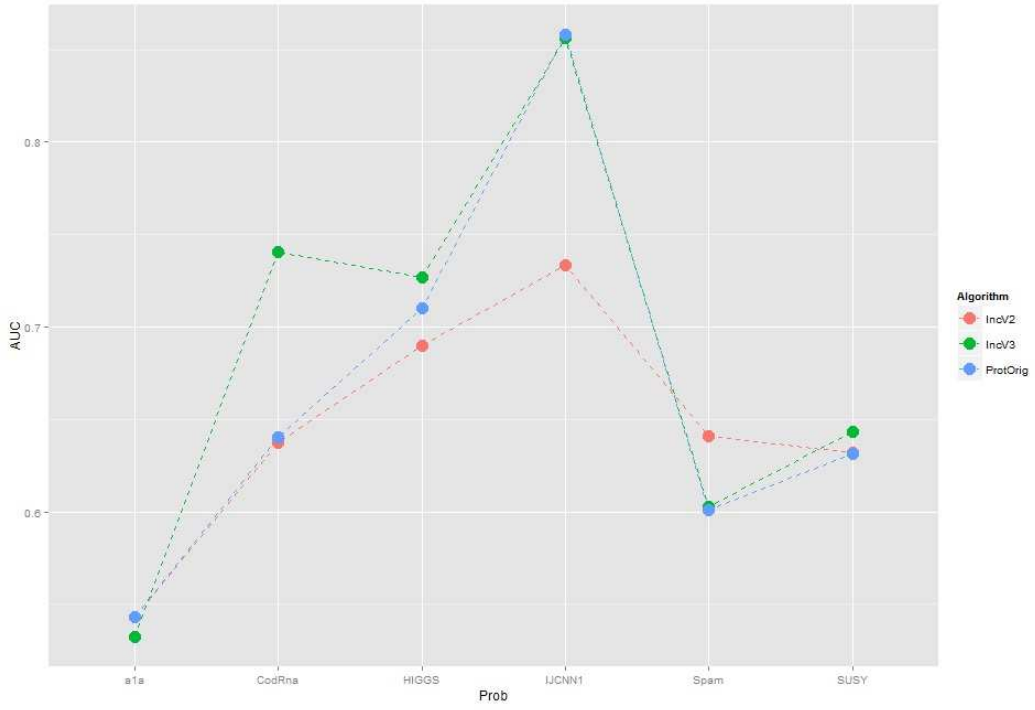

Figura 33 – Distribuição dos resultados dos algoritmos.

sendo τ_i o deslocamento, da média global, do algoritmo i. A hipótese nula diz que o desempenho dos algoritmos são estatisticamente iguais, já a alternativa implica que eles são diferentes.

O teste ANOVA retornou um *p-value* igual a 0,316, não rejeitando portanto a hipótese nula.

O teste de múltiplas comparações, novamente se tornaria desnecessário, uma vez que o ANOVA não detectou diferenças entre os algoritmos, porém, afim de verificar diferenças pontuais dos estimadores, foi realizado o teste de múltiplas comparações (Tukey) (MONTGOMERY, 2006), no teste Tukey é feito a comparação de todos algoritmos contra todos.

A Tabela 11 mostra o resultado da comparação. Nesta tabela o campo Estimativa é o valor pontual da diferença de desempenho entre os algoritmos, Inf. é o limite inferior do intervalo de confiança e Sup. é o limite superior do intervalo de confiança. Verifica-se que os intervalos de confiança englobam o valor 0, ou seja, confirmando o que o teste de ANOVA indicou. A Figura 34 ilustra o resultado da comparação de uma forma mais clara. Se analisado o estimador pontual, nota-se que o IncV3 foi um pouco superior aos demais.

Tabela 11 – Comparação Dunnet entre algoritmos.

Comparação	Estimativa	Inf.	Sup.
IncV2-IncV3	-0.04902	-0.13280	0.03476
ProtOrig-IncV3	-0.02810	-0.11188	0.05569
ProtOrig-IncV2	0.02092	-0.06286	0.10470

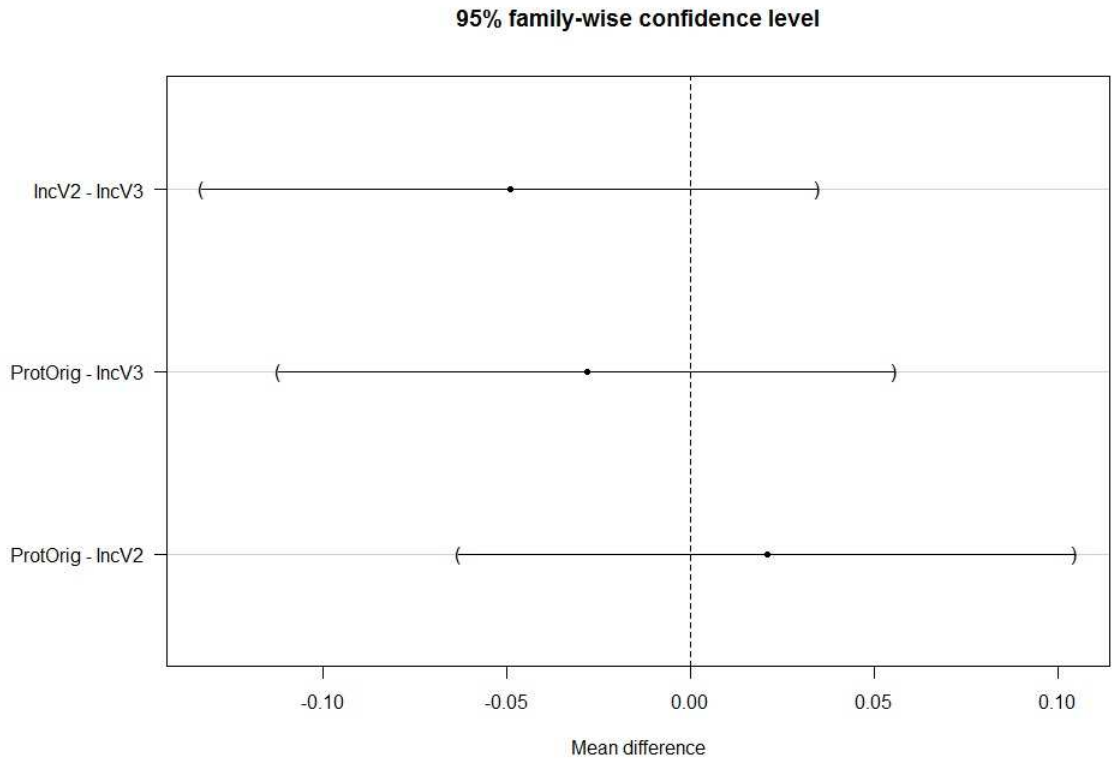

Figura 34 – Comparação Tukey entre os algoritmos.

5.5 Influência do tamanho de JT

Uma característica de todos algoritmos desenvolvidos, está na determinação do tamanho da janela JT. O tamanho de JT diz respeito ao desempenho do algoritmo quando a obtenção de um bom modelo preditor, e também, quanto ao custo computacional utilizado pelo algoritmo. Sabemos que JT muito grande pode impossibilitar o armazenamento de dados, inviabilizar a construção do GG, entre outros problemas. Janelas muito pequenas podem não armazenar informações mínimas para que o algoritmo realize o processo de filtragem de forma correta.

Dessa forma, os próximos tópicos serão destinados a investigar a influência do tamanho de JT, analisando a complexidade computacional e o desempenho do algoritmo.

5.5.1 Tempo Computacional

De acordo com Zhang e King (2002) a construção, com o algoritmo intuitivo, do Grafo de Gabriel tem complexidade computacional $O(n^3)$. Contudo, caso o Grafo de Gabriel seja construído a partir da estrutura da Triangulação de Delaunay, o Grafo de Gabriel pode, então ser construído com complexidade computacional $O(nlogn)$.

5.5.1.1 Complexidade computacional $O(n^3)$

Consideremos inicialmente a construção do Grafo de Gabriel (GG) com complexidade computacional $O(n^3)$. Neste caso, para n dados a máquina gastaria n^3 instruções para construção do GG. Seguindo essa ideia, consideremos um problema de n pontos em que os dados foram divididos em p janelas de tamanhos iguais (JT). Dessa forma, será necessário construir p GGs, em que cada GG possui n/p dados. Dessa maneira temos a seguinte equação para complexidade computacional:

$$p.\frac{n^3}{p^3} = \frac{n^3}{p^2} \tag{5.1}$$

Nota-se, claramente que a partição dos dados em janelas reduz a complexidade computacional do problema. Considerando um exemplo da base de dados *CodRna*, que possui 271.617 dados, a complexidade computacional para a construção do GG seria de:

$$n^3 = 271.617^3 = 2,0039\mathbf{x}10^{16}\ instruções \tag{5.2}$$

Considerando, agora a partição dos dados em 400 janelas de tamanho em torno de 680 dados por janela, tem-se que a complexidade computacional da construção do GG seria de:

$$\frac{271.617^3}{400^2} = 1,2524\mathbf{x}10^{11}\ instruções \tag{5.3}$$

A razão entre os tempos para as duas situações seria de $1,6\mathbf{x}10^5$. Supondo uma máquina que execute 10^6 instruções por segundo e considerando o primeiro caso o GG gastaria cerca de 231.930 dias, o que equivale a cerca de 635 anos para ser construído. Já o segundo caso, o GG gastaria em torno de 1,45 dias para ser construído.

Vale salientar que não foi considerado o tempo de treinamento para a construção do GG na Janela Geométrica (JG), sendo que a JG só existirá no caso em que houver partição dos dados. A não consideração se deve ao fato do não conhecimento do número de dados presentes em JG, possivelmente em termos de ordem de grandeza o tempo gasto para tratar JG pode ser desprezado.

5.5.1.2 Complexidade computacional $O(nlogn)$

Consideremos, agora, a construção do Grafo de Gabriel (GG) com complexidade computacional $O(nlogn)$. Neste caso, para n dados a máquina gastaria $O(nlogn)$ instruções para construção do GG. Seguindo essa ideia, consideremos um problema de n pontos em que os dados foram divididos em p janelas de tamanhos iguais (JT). Dessa forma, será necessário construir p GGs, em que cada GG possui n/p dados. Dessa maneira temos a seguinte equação para complexidade computacional:

$$p\frac{n}{p}log\left(\frac{n}{p}\right) = nlog\left(\frac{n}{p}\right) \tag{5.4}$$

Nota-se, claramente que a partição dos dados em janelas reduz a complexidade computacional do problema. Considerando um exemplo da base de dados *CodRna*, que possui 271.617 dados, a complexidade computacional para a construção do GG seria de:

$$nlog(n) = 271.617log(271.617) = 1,4759\mathbf{x}10^6\ instruções \tag{5.5}$$

Considerando, agora a partição dos dados em 400 janelas de tamanho em torno de 680 dados por janela, tem-se que a complexidade computacional da construção do GG seria de:

$$271.617log\left(\frac{271.617}{400}\right) = 7,691\mathbf{x}10^5\ instruções \tag{5.6}$$

A razão entre os tempos para as duas situações seria de $1,919$. Supondo uma máquina que execute 10^6 instruções por segundo e considerando o primeiro caso o GG gastaria cerca de 1,4759 segundos. Já o segundo caso, a construção dos 400 GGs gastaria em torno de 0,7691 segundos.

Vale salientar, novamente, que não foi considerado o tempo de treinamento para a construção do GG na Janela Geométrica (JG), sendo que a JG só existirá no caso em que houver partição dos dados. A não consideração se deve ao fato do não conhecimento do número de dados presentes em JG.

5.5.2 Desempenho ao classificar dados

A próxima análise consiste em verificar a eficiência do algoritmo IncV3 ao classificar dados considerando diferentes tamanhos de janelas.

Para realizar esta análise, será utilizado a base de dados *IJCNN1*. Vale salientar que não foram utilizadas outras bases devido a dificuldade em coletar dados, pois a variação do tamanho de JT e execução dos algoritmos demanda um tempo elevado. Dessa maneira, a análise que se segue pode não ser possível de extrapolar para outras bases.

5.5.2.0.1 Coleta de dados

Foram realizadas coletas com janelas em 5 tamanhos diferentes, com 100%, 50%, 10%, 1% e 0,1% dos dados. Para cada tamanho de janela diferente foram coletados 10 dados utilizando a técnica *prequential* e a acurácia.

Para avaliar a análise foi realizado um teste de comparação múltipla. O teste de hipóteses é definido abaixo com os requisitos de 95% de confiança e 75% de potência.

$$\begin{cases} H_0 : \tau_i = 0 \quad \forall \quad i \\ H_1 : \tau_i \neq 0 \end{cases}$$

sendo τ_i o deslocamento da média global do algoritmo i. A hipótese nula diz que o desempenho dos algoritmos são estatisticamente iguais, já a alternativa implica que eles são diferentes

O teste ANOVA retornou um *p-value* igual a $7,6x10^{11}$ o que implica em rejeitar a hipótese nula, ou seja o teste ANOVA indicou que existe influência na escolha do tamanho da janela. Nesse caso é necessário realizar o teste de comparação múltipla, a fim de verificar quais são as diferenças indicadas pelo teste ANOVA. A comparação realizada foi uma comparação entre todos contra todos (*Tukey*).

A Figura 35 mostra a distribuição dos dados de forma visual, nesta figura nota-se que, aparentemente, o desempenho do algoritmo para janela de tamanho 0,1% dos dados é inferior aos demais tamanhos.

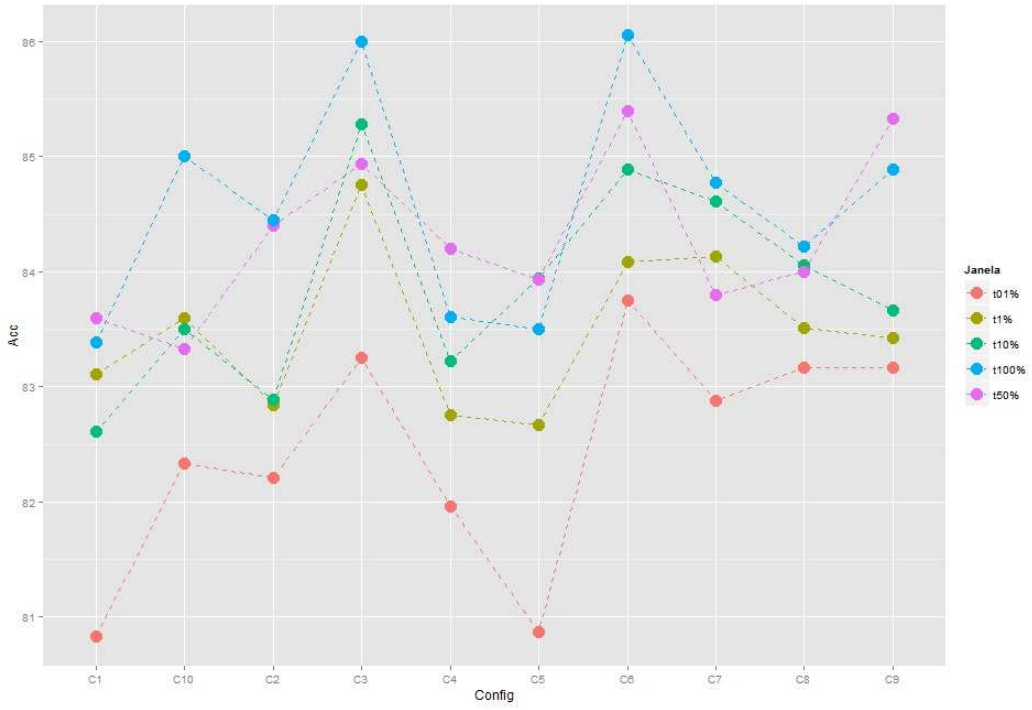

Figura 35 – Distribuição dos resultados para as diferentes janelas.

Tabela 12 – Teste estatístico comparação do tamanho das janelas.

Comparação	Estimativa	Inf.	Sup.
0,1%-100%	-0.025722	-0.033254	-0.018190
1%-100%	-0.013062	-0.020594	-0.005531
10%-100%	-0.008565	-0.016097	-0.001034
50%-100%	-0.003477	-0.011009	0.004055
1%-0,1%	0.012660	0.005128	0.020191
10%-0,1%	0.017157	0.009625	0.024688
50%-0,1%	0.022245	0.014713	0.029776
10%-1%	0.004497	-0.003035	0.012028
50%-1%	0.009585	0.002053	0.017117
50%-10%	0.005088	-0.002443	0.012620

Na Tabela 12 mostra o resultado da comparação. Nesta tabela o campo Estimativa é o valor pontual da diferença de desempenho entre os algoritmos, Inf. é o limite inferior do intervalo de confiança e Sup. é o limite superior do intervalo de confiança. Os resultados mostram que existe diferenças de desempenho na escolha do algoritmo, principalmente para janelas pequenas, como no caso do tamanho 0,1%, que foi inferior a todos os outros tamanhos de janela.

Como a escolha do tamanho da janela implica em vários fatores, como capacidade da memória, tempo computacional, velocidade do fluxo dos dados, entre outros, um teste de equivalência se torna mais apropriado. Neste teste verifica-se a equivalência entre dados coletados em um determinado intervalo aceitável, é como considerar diferenças dentro de certo intervalo desprezíveis.

Nesse caso, consideramos o intervalo de diferença entre -0,02 a 0,02 desprezível. Lembrando que em termos de acurácia, que varia de 0 a 100, um intervalo de tamanho 0,04 é um intervalo pequeno.

A Figura 36 mostra o teste de equivalência, nesta figura notamos que foram detectadas diferenças de desempenho entre os tamanhos das janelas na comparação entre 0,1% e 100%, 0,1% e 10%, 0,1% e 50%, além de uma pequena diferença na comparação entre 1% e 100% e 0,1% e 1%. Este resultado mostra que a escolha de janelas pequenas, como neste caso em particular de tamanho 0,1% do total dos dados resulta em perda de desempenho. Já os demais tamanhos se mostraram equivalentes, isso implica que o tamanho da janela, possivelmente, não é um parâmetro que determine o desempenho do classificador incremental.

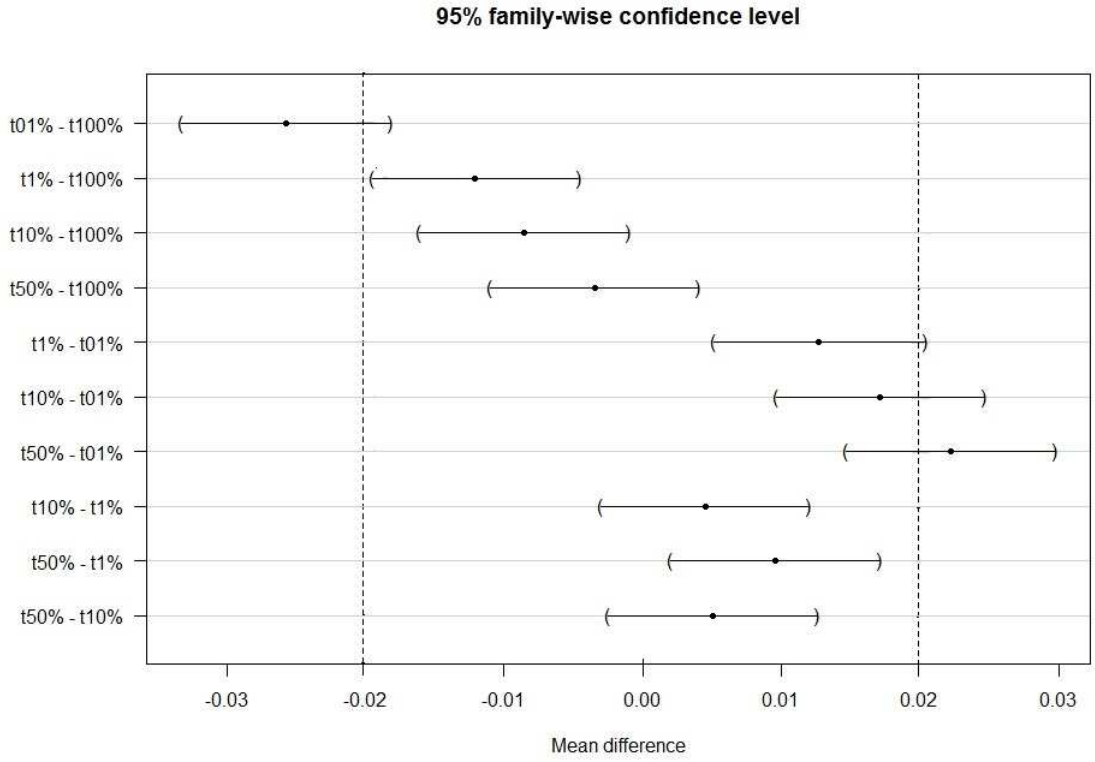

Figura 36 – Teste de equivalência entre as janelas.

6 Conclusões e Propostas de Continuidade

Essa dissertação abordou ideias e trabalhos presentes na literatura para desenvolver algoritmos incrementais baseados na estrutura geométrica do grafo de gabriel e utilizando o classificador *Chip-clas* (TORRES et al., 2015), foram implementados quatro algoritmos distintos, sendo que um deles possui um mecanismo explícito de detecção de mudança. A utilização do Chip-clas resultou em uma importante característica para os algoritmos desenvolvidos, pois o classificador não necessita da seleção de parâmetros por um especialista. Esta característica é importante pois algoritmos incrementais recorrem a todo momento ao classificador e a determinação de parâmetros é uma tarefa relativamente custosa para o algoritmo.

Os algoritmos foram desenvolvidos pensando no cenário *data stream*, onde os dados chegam em pacotes com uma componente temporal associada. Os algoritmos se baseiam na estrutura do grafo de gabriel para selecionar e excluir dados e armazená-los em memória, sendo assim, definidos como algoritmos incrementais com memória parcial.

O primeiro algoritmo, denominado IncV0, foi avaliado utilizando 3 bases de dados e comparado com a metodologia tradicional, que assume que todos os dados encontram-se disponíveis a priori. Esse teste teve como intuito, verificar se para diferentes bases os algoritmos são equivalentes. Dessa forma, realizou-se 3 testes estatísticos, sendo que em todos os 3 testes os algoritmos se mostraram equivalentes. Com isso o IncV0 se mostrou equivalente ao treinamento tradicional para as bases de dados consideradas. Este teste foi importante para validar o algoritmo e permitir a continuação da pesquisa da implementação destes métodos.

Os segundo e terceiro algoritmos, denominados IncV1 e IncV2, respectivamente, foram testados comparando com outras técnicas, IncV0, Batch e SVM (disponível na literatura). Foi realizado um teste estatístico de comparação múltipla, onde os algoritmos se mostraram equivalentes, porém ao analisar o estimador pontual, o IncV2 se mostrou ligeiramente superior aos demais. Essa ligeira diferença é importante, pois ao utilizar bases maiores, os intervalos de confiança tendem a se encurtar e pode resultar em uma diferença estatística entre os algoritmos. Outra importante informação foi a equivalência das metodologias com relação ao SVM, uma vez que a utilização das SVMs implica no ajuste de parâmetros por um especialista. Para este teste, os parâmetros foram ajustados com a utilização da validação cruzada, sendo que a cada avaliação de dados, foi necessário realizar a validação cruzada. O fato do Chip-clas não necessitar de ajuste de parâmetros faz com que sua utilização possa ser preferida em relação as SVMs, especialmente, no cenário de *data stream*, onde decisões precisam ser tomadas com extrema rapidez.

O algoritmo IncV3 foi comparado com o IncV2, através de um teste estatístico com 11 bases distintas. Os resultados mostram a equivalência entre os algoritmos.

Além disso, os algoritmos IncV2 e IncV3 foram comparados com o algoritmo desenvolvido por Zheng et al. (2010), chamado, aqui, de ProtOrig. Para isso foram utilizadas 6 bases de dados com grande volume de dados. Os testes realizados mostram a equivalência das metodologias. Vale salientar que o ProtOrig necessita definir parâmetros da SVM, além de mais dois parâmetros do algoritmo. Por sua vez, o IncV3 necessita da definição de dois parâmetros e o IncV2 não necessita da definição de parâmetros.

A última análise realizada foi com relação a influência do tamanho da janela JT. Este tamanho esta relacionado a diversos fatores, como memória disponível, tempo computacional, velocidade do fluxo de dados, entre outros aspectos. A análise do tempo computacional, mostrou que quanto menor o tamanho de JT, menor o tempo computacional gasto para gerar os grafos de gabriel. Uma segunda análise foi realizada em termos do desempenho ao classificar dados. Esta análise foi realizada com uma única base, onde foram realizados testes com diferentes tamanhos de janela. Como resultado da análise, um teste estatístico de equivalência, mostrou que, para essa base a escolha de janelas pequenas (menor que 1% dos dados) resultam em uma queda de desempenho do classificador, mas janelas um pouco maiores (maiores ou iguais a 1% dos dados) resultam em desempenho equivalente a janelas grandes. Essa conclusão não pode ser generalizada para outras bases, mas traz uma suspeita de que a escolha do tamanho de JT não seja um parâmetro decisivo no desempenho do algoritmo.

Por fim, essa dissertação apresentou novas ferramentas para o tratamento de dados que necessitam ser trabalhados de forma incremental, através de algoritmos incrementais que utilizam informações estruturais dos dados para obtenção de memórias parciais e obtenção dos classificadores. Os resultados mostram a eficiência das técnicas implementadas, bem como suas vantagens relacionadas a algumas técnicas presentes na literatura. Espera-se que estas ferramentas desenvolvidas possam ser aproveitadas em trabalhos futuros e problemas reais.

6.1 Propostas de Continuidade

Sugere-se algumas propostas de continuidade do trabalho:

- Estudo da otimização dos parâmetros de quarentena do IncV3.
- Desenvolvimento de um algoritmo online incremental, que utilize informações geométricas, dados distantes e erros de classificação.
- Construção online do Grafo de Gabriel, inserção de novos dados ao grafo sem necessitar de uma completa reconstrução do mesmo.
- Análise estatística, com diversas bases, da influência do tamanho de JT no desempenho dos algoritmos ao classificar dados

Referências

ADE, R. R.; DESHMUKH, P. R. Methods for incremental learning: A survey. *International Journal of Data Mining Knowledge Management Process (IJDKP)*, v. 3, n. 4, p. 119–125, 2013.

AGRAWAL, D.; DAS, S.; ABBADI, A. E. Big data and cloud computing: Current state and future opportunities. In: *Proceedings of the 14th International Conference on Extending Database Technology*. New York, NY, USA: ACM, 2011. (EDBT/ICDT '11), p. 530–533. ISBN 978-1-4503-0528-0. Disponível em: ⟨http://doi.acm.org/10.1145/1951365.1951432⟩.

AUPETIT, M.; CATZ, T. High-dimensional labeled data analysis with topology representing graphs. *Neurocomput.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 63, p. 139–169, jan. 2005. ISSN 0925-2312. Disponível em: ⟨http://dx.doi.org/10.1016/j.neucom.2004.04.009⟩.

BARROSO, M. M. A. Operações elementares em grafos. *EMARC, SBMAC*, v. 7, 2007.

BENNETT, K. P.; BREDENSTEINER, E. J. Duality and geometry in svm classifiers. In: *In Proc. 17th International Conf. on Machine Learning*. [S.l.]: Morgan Kaufmann, 2000. p. 57–64.

BERG, M. d. et al. *Computational Geometry: Algorithms and Applications*. 3rd ed.. ed. Santa Clara, CA, USA: Springer-Verlag TELOS, 2008. ISBN 3540779736, 9783540779735.

BURGES, C. J. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, Springer, v. 2, n. 2, p. 121–167, 1998.

DURAN, J. L. M.; PEREZ, S. O. Diagramas de voronoi de alcance limitado. *International Joint Conference on Neural Networks. IJCNN Killarney*, 2015.

FAWCETT, T. An introduction to roc analysis. pattern recogn. lett. *27*, p. 861–874, 2006.

FISHER, D. H. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, n. 2, p. 139–172, 1987.

GAMA, J. a. et al. A survey on concept drift adaptation. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 46, n. 4, p. 44:1–44:37, mar. 2014. ISSN 0360-0300. Disponível em: ⟨http://doi.acm.org/10.1145/2523813⟩.

GIRAUD-CARRIER, C. A note on the utility of incremental learning. *Ai Communications*, IOS Press, v. 13, n. 4, p. 215–223, 2000.

GORMAN, R. P.; SEJNOWSKI, T. J. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks, Vol. 1, pp. 75-89*, 1988.

HETTICH, S.; BLAKE, C. L.; MERZ, C. J. Uci repository of machine learning databases. *Department of Information and Computer Science,University of California, Irvine, CA,*, 1998. Disponível em: ⟨http://www.ics.uci.edu/mlearn/MLRepository.html⟩.

LASKOV, P. et al. Incremental support vector learning: Analysis, implementation and applications. *J. Mach. Learn. Res.*, JMLR.org, v. 7, p. 1909–1936, dez. 2006. ISSN 1532-4435. Disponível em: ⟨http://dl.acm.org/citation.cfm?id=1248547.1248616⟩.

LI, J.; KUO, C.-C. J. *A dual graph approach to 3D triangular mesh compression*. [S.l.]: Proceedings of the International Conference on Image Processing, 1998a. v. 2. 891–894 p.

LICHMAN, M. Uci machine learning repository. *Irvine, CA: University of California, School of Information and Computer Science*, 2013. Disponível em: ⟨http://archive.ics.uci.edu/ml⟩.

LITTLE, M. et al. Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. *BioMedical Engineering OnLine*, 2007.

MALOOF, M. A.; MICHALSKI, R. S. Selecting examples for partial memory learning. *Machine Learning*, Kluwer Academic Publishers, Boston, v. 41, n. 1, p. 27–52, 2000.

MONTGOMERY, D. C. *Design and Analysis of Experiments*. [S.l.]: John Wiley & Sons, 2006. ISBN 0470088109.

OKABE, A.; BOOTS, B.; SUGIHARA, K. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. New York, NY, USA: John Wiley & Sons, Inc., 1992. ISBN 0-471-93430-5.

PINTO, C. M. S. *Algoritmos Incrementais para Aprendizagem Bayesiana*. 111 p. Tese (Doutorado) — Departamento de Ciência de Computadores, Faculdade de Economia da Universidade do Porto, 2005.

POLIKAR, R. et al. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on System, Man and Cybernetics (C), Special Issue on Knowledge Management*, v. 31, p. 497–508, 2001.

PRATI, R. C.; BATISTA, G. E. A. P. A.; MONARD, M. C. Curvas roc para avaliação de classificadores. *Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo*, 2010.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2008. ISBN 3-900051-07-0. Disponível em: ⟨http://www.R-project.org⟩.

ROSSI, A. L. D. Meta-aprendizado aplicado a fluxos contínuos de dados. tese (doutorado). *Universidade de São Paulo, São Paulo*, 2014.

SCHAPIRE, R. E. A brief introduction to boosting. In: *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. (IJCAI'99), p. 1401–1406. Disponível em: ⟨http://dl.acm.org/citation.cfm?id=1624312.1624417⟩.

SYED, N.; LIU, H.; SUNG, K. K. Incremental learning with support vector machines. *Proceedings of the workshop on support vector machines at the international joint conference on artificial intelligence (IJCAI-99)*, Stockholm, Sweden, n. 2, 1999.

TORRES, L. C.; CASTRO, C. L.; BRAGA, A. P. Gabriel graph for dataset structure and large margin classification: A bayesian approach. *ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium)*, i6doc.com, p. 237–242, 2015. Disponível em: ⟨http://www.i6doc.com/en/⟩.

TORRES, L. C. B. *Uma Nova Abordagem Baseada em Margem para Seleção de Modelos Neurais*. 52 p. Dissertação (Mestrado) — Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, 2012.

TORRES, L. C. B. *Classificador por Arestas de Suporte (clas): Métodos de Aprendizado Baseados em Grafos de GabrieL*. 108 p. Tese (Doutorado) — Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, 2016.

TORRES, L. C. B. et al. Distance-based large margin classifier suitable for integrated circuit implementation. *Electronics Letters*, v. 51, n. 24, p. 1967–1969, 2015. ISSN 0013-5194.

YOSHIDA, M. L. *Aprendizado Supervisionado Incremental de Redes Bayesianas para Mineração de Dados*. 132 p. Dissertação (Mestrado) — Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, 2007.

ZHANG, W.; KING, I. A study of the relationship between support vector machine and gabriel graph. In: IEEE. *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*. [S.l.], 2002. v. 1, p. 239–244.

ZHENG, J. et al. An online incremental learning support vector machine for large-scale data. In: DIAMANTARAS, K. I.; DUCH, W.; ILIADIS, L. S. (Ed.). *ICANN (2)*. Springer, 2010. (Lecture Notes in Computer Science, v. 6353), p. 76–81. ISBN 978-3-642-15821-6. Disponível em: ⟨http://dblp.uni-trier.de/db/conf/icann/icann2010-2.html#ZhengYSZ10⟩.