Ensembles na classificação relacional

Nils Ever Murrugarra Llerena

Ensembles na classificação relacional

Nils Ever Murrugarra Llerena

Orientador: *Prof. Dr. Alneu de Andrade Lopes*

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Ciências de Computação e Matemática Computacional. VERSÃO REVISADA

USP - São Carlos
Setembro/2011

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

M979e

Murrugarra-Llerena, Nils Ever
Ensembles na Classificação Relacional / Nils Ever Murrugarra-Llerena; orientador Alneu de Andrade Lopes -- São Carlos, 2011.
105 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2011.

1. Aprendizado de Máquina. 2. Ensembles. 3. Classificadores relacionais baseados em grafos. 4. Boosting. 5. Bagging. I. de Andrade Lopes, Alneu, orient. II. Título.

Dedicatória

Aos meus pais,
Salomon e Genny.
E aos meus irmãos,
Hans, Maisi e Jeffri.

Agradecimentos

Gostaria de agradecer primeiramente a minha família, motivo pelo qual os primeiros parágrafos estão em espanhol.

A mis queridos padres, Salomon y Genny, por su apoyo, cariño, dedicación y comprensión. Especialmente, por ser mi fuente de inspiración para cada dia ser una mejor persona.

A mis estimados hermanos, Hans, Maisi y Jeffri, por los momentos amenos compartidos y por todo su apoyo durante esta etapa. Adicionalmente, a mi prima Sandra por sus frases de aliento en momentos dificiles.

Ao professor Alneu, pela dedicação, paciência e trabalho conjunto. Especialmente, pela motivação e incentivo dado durante todo este tempo.

Aos amigos e professores do LABIC, pelo companheirismo e pelos bons momentos de convivência, dentro e fora do laboratório: Bruno, Victor, Everton, Igor, Robson, Rafael, Ricardo, Fabiano, Tatiane, Mel, Nathalie, Jean, Marcos, Alinson, Diego, Jorge, Pedro, Newton, Celso, Diogo, Gustavo, Carolina e Solange.

Aos amigos do SECC, por ser uma fonte de inspiração na minha vida e por ser um dos motivos para que eu tenha feito o mestrado no Brasil. Gostaria de agradecer especialmente a: Marks Calderón, Ivan Sipirán, David Wong, Jorge Guevara, Eduardo Rodriguez, Fredy Carranza, Jorge Valverde, Pedro Shiguihara e Luis Lujan.

Aos amigos peruanos no ICMC, que são como minha família em São Carlos. Gostaria de agradecer especialmente a: Jean Huertas, Christian Wong, Fernando Alva, Laura Florian, Marleny Hilasaca e Ivar Vargas.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pelo apoio financeiro.

E a todos, que direta ou indiretamente, contribuíram para a realização deste trabalho.

Resumo

Em diversos domínios, além das informações sobre os objetos ou entidades que os compõem, existem, também, informações a respeito das relações entre esses objetos. Alguns desses domínios são, por exemplo, as redes de co-autoria, e as páginas Web. Nesse sentido, é natural procurar por técnicas de classificação que levem em conta estas informações. Dentre essas técnicas estão as denominadas classificação baseada em grafos, que visam classificar os exemplos levando em conta as relações existentes entre eles. Este trabalho aborda o desenvolvimento de métodos para melhorar o desempenho de classificadores baseados em grafos utilizando estratégias de *ensembles*. Um classificador *ensemble* considera um conjunto de classificadores cujas predições individuais são combinadas de alguma forma. Este classificador normalmente apresenta um melhor desempenho do que seus classificadores individualmente. Assim, foram desenvolvidas três técnicas: a primeira para dados originalmente no formato proposicional e transformados para formato relacional baseado em grafo e a segunda e terceira para dados originalmente já no formato de grafo. A primeira técnica, inspirada no algoritmo de *boosting*, originou o algoritmo KNN Adaptativo Baseado em Grafos (A-KNN). A segunda técnica, inspirada no algoritmo de *Bagging* originou três abordagens de *Bagging* Baseado em Grafos (BG). Finalmente, a terceira técnica, inspirada no algoritmo de *Cross-Validated Committees*, originou o *Cross-Validated Committees* Baseado em Grafos (CVCG). Os experimentos foram realizados em 38 conjuntos de dados, sendo 22 conjuntos proposicionais e 16 conjuntos no formato relacional. Na avaliação foi utilizado o esquema de *10-fold stratified cross-validation* e para determinar diferenças estatísticas entre classificadores foi utilizado o método proposto por Demšar (2006). Em relação aos resultados, as três técnicas melhoraram ou mantiveram o desempenho dos classificadores bases. Concluindo, *ensembles* aplicados em classificadores baseados em grafos apresentam bons resultados no desempenho destes.

Abstract

In many fields, besides information about the objects or entities that compose them, there is also information about the relationships between objects. Some of these fields are, for example, co-authorship networks and Web pages. Therefore, it is natural to search for classification techniques that take into account this information. Among these techniques are the so-called graph-based classification, which seek to classify examples taking into account the relationships between them. This paper presents the development of methods to improve the performance of graph-based classifiers by using strategies of ensembles. An ensemble classifier considers a set of classifiers whose individual predictions are combined in some way. This combined classifier usually performs better than its individual classifiers. Three techniques have been developed: the first applied for originally propositional data transformed to relational format based on graphs and the second and the third applied for data originally in graph format. The first technique, inspired by the boosting algorithm originated the Adaptive Graph-Based K-Nearest Neighbor (A-KNN). The second technique, inspired by the bagging algorithm led to three approaches of Graph-Based Bagging (BG). Finally the third technique, inspired by the Cross-Validated Committees algorithm led to the Graph-Based Cross-Validated Committees (CVCG). The experiments were performed on 38 data sets, 22 datasets in propositional format and 16 in relational format. Evaluation was performed using the scheme of 10-fold stratified cross-validation and to determine statistical differences between the classifiers it was used the method proposed by Demšar (2006). Regarding the results, these three techniques improved or at least maintain the performance of the base classifiers. In conclusion, ensembles applied to graph-based classifiers have good results in the performance of them.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

CAPÍTULO

1

Introdução

1.1 Contexto e Motivação

O Aprendizado de Máquina (AM) é um campo de pesquisa em Inteligência Artificial que está concentrado no desenvolvimento de algoritmos que melhoram automaticamente algum aspecto de seu desempenho através da experiência (Mitchell, 1997; Alpaydin, 2004). Além disso, AM apresenta aplicações bem sucedidas em inúmeras áreas, entre as quais pode-se citar visão computacional, processamento de linguagem natural, diagnóstico médico, bioinformática, reconhecimento de padrões, entre outras.

Uma das tarefas de AM é a classificação, a qual consiste na generalização de experiências passadas para predizer algum valor discreto (conhecido como classe) em novos exemplos (Hastie et al., 2003). Na tarefa de classificação, existem as técnicas de *ensembles*, cujo objetivo é melhorar a classificação combinando classificadores. Estas técnicas consideram a estratégia utilizada quando uma pessoa precisa tomar uma decisão difícil, ela geralmente prefere considerar opiniões de vários especialistas que considerar a opinião de apenas um. O mapeamento de tal estratégia na área de aprendizado de máquina considera que um modelo induzido por um algoritmo pode ser considerado como uma opinião. Assim, a idéia principal por trás da técnica de *ensemble* é combinar classificadores individuais para obter um classificador que supera a cada um deles isolados. Estudos sobre esta técnica têm levado a uma área ativa de pesquisa em aprendizado de máquina com aplicações bem sucedidas em

muitos campos, como finanças (Leigh et al., 2002), bioinformática (Tan e Gilbert, 2003), quimioinformática (Merkwirth et al., 2004), medicina (Mangiameli et al., 2004), recuperação de imagens (Lin et al., 2006), fabricação (Maimon e Rokac, 2004), geografia (Bruzzone et al., 2004), entre outras.

1.2 Representação dos Dados

Neste trabalho são apresentadas técnicas *ensemble* para diferentes representações de dados. Via de regra dados ou informação de entradas dos algoritmos de AM são representados em tabelas ou vetores atributo-valor. Tal formato não representa naturalmente relações entre objetos ou entidades. Nesse último caso, representações relacionais, mais expressivas, descrevem com mais naturalidade os dados. Essas descrições são formalizadas a seguir.

1.2.1 Representação Proposicional em Tabelas Atributo-Valor

uma representação proposicional dos dados, por meio de uma tabela atributo-valor, descreve apenas os valores das características individuais dos objetos. A Tabela 1.1 exemplifica a descrição atributo-valor, considerando um conjunto de exemplos E com n exemplos e m atributos. Nessa tabela, a linha i refere-se ao i-ésimo exemplo ($i = 1, 2, ..., n$) e a entrada x_{ij} refere-se ao valor do j-ésimo ($j = 1, 2, ..., m$) atributo X_j do exemplo i.

Como pode ser notado, exemplos são tuplas $E_i = (x_{i1}, x_{i2}, ..., x_{im}, y_i) = (\vec{x_i}, y_i)$, também denotados por (x_i, y_i), onde fica subentendido o fato que x_i é um vetor. A última coluna, Y, contém o atributo meta, também chamado de classe. Observa-se que cada x_i é um elemento do conjunto $dom(X_1) \times dom(X_2) \times ... \times dom(X_m)$, onde $dom(X_j)$ é o domínio do atributo X_j e y_i pertence a uma das k classes, isto é, $y_i \in \{c_1, c_2, ..., c_k\}$.

	X_1	X_2	$\cdots$	X_m	Y
E_1	x_{11}	x_{12}	$\cdots$	x_{1m}	y_1
E_2	x_{21}	x_{22}	$\cdots$	x_{2m}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
E_n	x_{n1}	x_{n2}	$\cdots$	x_{nm}	y_n

Tabela 1.1: Conjunto de exemplos no formato atributo-valor.

Usualmente, um conjunto de exemplos é dividido em dois subconjuntos, o conjunto de treino, denominado E_{tr}, usado para o aprendizado do conceito e o

conjunto de teste, denominado E_{ts}, usado para medir o grau de efetividade do conceito aprendido. Estes subconjuntos são normalmente disjuntos.

1.2.2 Representação Relacional

Considerando que a representação proposicional descreve atributos individuais dos objetos, ela não pode "elegantemente" representar domínios envolvendo várias entidades, bem como as relações entre elas. Um exemplo deste domínio, são as redes sociais, em que as pessoas são as entidades e suas interações sociais são dadas por suas conexões. Vários pesquisadores da área, como Michalski (1983) e Plotkin (1970), perceberam estas limitações e começaram a empregar novas representações de conhecimento mais expressivas. Essas pesquisas centraram-se no uso de *frameworks* capazes de representar um número variável de entidades ou objetos, bem como as relações existentes entre elas. Estas representações são denominadas relacionais, e uma das possibilidades para representar estas informações é via grafos. Representações relacionais derivadas da lógica de primeira ordem são chamadas representações lógicas e não são abordadas neste trabalho. Neste trabalho quando se refere à representação ou a classificadores relacionais, refere-se à representação relacional baseada em grafos e aos classificadores baseados em grafos.

Observe também que no trabalho utiliza-se dados que ja são originariamente relacionais, isto é, representados em grafos e, também, dados atributo-valor representados em grafos usando relações de similaridade entre os objetos, comentadas à frente.

Dados Originais em formato Relacional

Neste trabalho, dados em formato relacional são representados por meio de grafos. Segundo De Raedt (2008), a representação baseada em grafos apresenta uma descrição mais expressiva, representando os atributos dos objetos em conjunto com as relações existentes entre eles. Um grafo é definido como uma estrutura $G(V, A)$, onde V é um conjunto não vazio de objetos denominados vértices e A é um conjunto de arestas, em que A representa uma relação em $V \times V$. Por exemplo, na Figura 1.1 é apresentado um grafo de relações de co-autoria, sendo os vértices os autores e as arestas representando se os autores trabalharam juntos em um artigo.

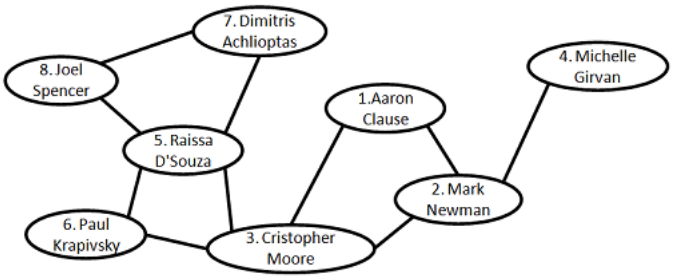


Figura 1.1: Exemplo de um grafo de co-autoria de artigos, adaptado de Motta (2009)

Dados Originais em formato Tabela Atributo-Valor

Observa-se, entretanto, que é possível construir uma representação via grafos para dados originariamente representados em uma tabela atributo-valor. Neste caso ao invés de representar os objetos por seus atributos em uma tabela de exemplos, passa-se a representar apenas as relações entre objetos. No caso, a relação considerada é uma relação de similaridade com os vértices representando os objetos. As abordagens comuns para construção de grafos a partir de uma representação por tabela atributo-valor são aquelas baseadas em relações de vizinhança entre exemplos. No grafo gerado, vértices representam exemplos (linhas da tabela) e arestas relações de similaridade ou vizinhança entre exemplos. A seguir, são descritas três dessas abordagens.

Redes KNN (Zhu, 2005) Redes KNN são grafos, onde o parâmetro K é usado para especificar o número de vértices vizinhos para se conectar. Qualquer vértice v_i irá se conectar com seus K vizinhos usando uma aresta não pesada e não dirigida.

Redes ε (Belkin e Niyogi, 2003) Redes ε são representadas por meio de grafos, nos quais as conexões entre pares de vértices são estabelecidas considerando um raio ε como limiar. Qualquer par de vértices v_i e v_j com a distância $d(v_i, v_j) < \varepsilon$ são conectados.

Redes baseadas em similaridade Considera-se aqui uma rede baseada em similaridade como uma variante da rede KNN, com arestas dirigidas nas conexões dos vértices.

1.3 Identificação do Problema

Considerando as pesquisas apontadas na subseção 1.1. Um dos principais resultados é que o *ensemble* geralmente apresenta um melhor desempenho que os classificadores individuais que o originaram (Dietterich, 2000b). Adicionalmente, a acurácia e a diversidade de classificadores individuais são fatores importantes na acurácia do modelo combinado (Hansen e Salamon, 1990b). No entanto, na medida do nosso conhecimento, nenhum estudo anterior avaliou técnicas de *ensemble* aplicada no contexto de classificadores relacionais baseados em grafos ou simplesmente classificadores relacionais. Embora exista estudos de *ensembles* de teorias ou clausulas de primeira ordem no contexto de Programação Lógica Indutiva (ILP) (Goadrich e of Wisconsin Madison, 2007). Visto que dados no formato de grafo são muito comuns, a pesquisa sobre técnicas para melhorar classificadores relacionais baseados em grafos são certamente pertinentes.

Algumas das aplicações de classificação baseada em grafos são para classificação de patentes (Chakrabarti et al., 1998), trabalhos de pesquisa científica (Taskar et al., 2001) e páginas *Web* (Chakrabarti et al., 1998; Neville e Jensen, 2003). Além dessas aplicações, há aplicações em detecção de fraudes, detecção de casos de terrorismo, *marketing*, entre outras. Em relação à detecção de fraudes no trabalho de Fawcett et al. (1997) são apresentadas diversas técnicas de detecção de fraudes usando uma rede de ligações entre pessoas. No trabalho de Cortes et al. (2001) é apresentada uma aplicação de detecção de fraudes, usando uma rede de telecomunicações. Em relação à detecção de casos de terrorismo, uma aplicação neste tópico foi desenvolvida pela Agência de Segurança Nacional (*National Security Agency*) dos Estados Unidos (Tumulty, 2006). essa agência decidiu criar uma base de dados de ligações telefônicas feitas nos Estados Unidos. Essa aplicação considera que pessoas tendem a associar-se com outras de interesses ou características similares (*guilty by association*) (Galstyan e Cohen, 2005; Macskassy e Provost, 2005). Do mesmo modo, na área de *marketing*, os clientes podem ser representados em uma rede baseada em produtos que compram. Essa rede pode ser usada para fazer recomendações de produtos como é mostrado em Domingos e Richardson (2001). Concluindo, a classificação baseada em grafos pode ser aplicada em diferentes domínios e problemas da vida real e as técnicas de *ensemble* que foram bem sucedidas na melhoria de classificadores proposicionais devem ser investigadas no âmbito de classificadores relacionais.

1.4 Objetivos e Metodologia

O objetivo geral deste trabalho é o desenvolvimento de *ensembles* para classificadores relacionais baseados em grafos. Sendo assim, os objetivos específicos são descritos a seguir.

- Explorar técnicas baseadas em similaridade para construção de uma representação relacional a partir de dados proposicionais, com os vértices dos grafos representando os objetos e as arestas representando relações de similaridade entre os objetos.
- Adaptar técnicas de amostragem para dados representados no formato de grafo.
- Investigar o mapeamento de técnicas proposicionais para *boosting*, *bagging* e *ensembles* para técnicas relacionais correlatas.
- Entender o comportamento das técnicas de amostragem e determinar se geram diversidade nos classificadores do *ensemble*.
- Avaliar as técnicas propostas.

Os experimentos foram realizados utilizando 38 conjuntos de dados, sendo 22 conjuntos originalmente proposicionais, representados por tabelas atributo-valor, e 16 conjuntos originalmente relacionais, representados por grafos. Para visualização dos grafos foi utilizada a ferramenta PEx (Paulovich et al., 2007) com algumas adaptações, auxiliando no entendimento do comportamento dos algoritmos.

Para avaliação dos grafos construídos a partir de uma tabela atributo-valor (detalhado na Seção 5.1.1), foi utilizado o esquema de *10-fold stratified cross-validation* em 3 versões dos dados: o conjunto original, o conjunto com 5% e com 10% de ruído. A adição de ruídos deveu-se ao fato conhecido e discutido na literatura (Dietterich, 2000b) de que principalmente a técnica de boosting pode deteriorar-se quando aplicada em dados com ruídos.

Foram feitas comparações com os algoritmos: K-Nearest Neighbor, C4.5, Naive Bayes e SVM; considerando suas versões isoladas e usando *boosting*. Para avaliação em conjuntos relacionais, foi utilizado o esquema de *10-fold stratified cross-validation* adaptado para grafos (detalhado na Seção 5.1.2). Nesta avaliação, foram comparados os classificadores relacionais base com os *ensembles* desenvolvidos.

Nas duas formas de avaliação, foi utilizado o método descrito em Demšar (2006). Considerando este método, os algoritmos propostos ou melhoraram ou mantiveram o desempenho dos classificadores bases, em nenhuma situação houve piora.

1.5 Contribuições

Neste trabalho, foram desenvolvidas técnicas *ensembles* baseadas em grafos tanto para dados originalmente no formato de grafos como para dados proposicionais.

Para dados originalmente no formato proposicional foi desenvolvido:

- O algoritmo KNN Adaptativo Baseado em Grafos (A-KNN), inspirado na técnica de *boosting*.

Para dados no formato relacional:

- *Bagging* baseado em Grafos (BG), e
- *Cross-Validated Committees* baseados em Grafos (CVCG).

Adicionalmente, no contexto relacional, foi realizado um estudo da diversidade dos classificadores gerados e sua importância no desempenho do *ensemble*.

Das três técnicas de *ensemble* propostas, duas foram aceitas para publicação em um importante *workshop* da área, parte da *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2011)*. No CoLISD (*Collective Learning and Inference on Structured Data*) foram aceitos os artigos:

1. Murrugarra-Llerena, Nils e de Andrade Lopes, Alneu (2011). *A Graph-based Bagging*. Em *Workshop*: *Collective Learning and Inference on Structured Data* (CoLISD). *Proceedings* da *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases* (ECML-PKDD 2011). Atenas, Grécia (Murrugarra-Llerena e de Andrade Lopes, 2011b).

2. Murrugarra-Llerena, Nils e de Andrade Lopes, Alneu (2011). *An Adaptive Graph-Based K-Nearest Neighbor*. Em *Workshop*: *Collective Learning and*

Inference on Structured Data (CoLISD). *Proceedings* da *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases* (ECML-PKDD 2011). Atenas, Grécia (Murrugarra-Llerena e de Andrade Lopes, 2011a).

Finalmente, foram desenvolvidas e integradas algumas caracteristicas na ferramenta PEx (Paulovich et al., 2007) para melhorar o entendimento do comportamento dos *ensembles*. Seja no contexto proposicional como relacional.

1.6 Organização da monografia

O restante deste trabalho está organizado da seguinte forma: o próximo capítulo aborda os principais conceitos, métodos de construção e técnicas *ensembles*. O Capítulo 3 descreve o tópico de classificação baseada em grafos, seus principais conceitos e algoritmos utilizados. A proposta de pesquisa relativa à *ensembles* na classificação relacional é apresentada no Capítulo 4. A seguir, no Capítulo 5, é descrita a avaliação experimental e os resultados alcançados. Em seguida, o Capítulo 6 apresenta uma discussão dos resultados obtidos. Finalmente, no Capítulo 7, são apresentadas as conclusões e trabalhos futuros.

CAPÍTULO

2

Classificadores *Ensembles*

Classificadores *Ensembles* são algoritmos de aprendizado supervisionado que constroem um conjunto de modelos e classificam novos exemplos usando uma votação das predições dos vários modelos (Dietterich, 2000a). Neste capítulo é apresentada a técnica de *ensemble*, seus principais conceitos e algoritmos. Observa-se que não foram encontradas na revisão da literatura da área técnicas de *ensembles* aplicadas a classificadores baseados em grafos. Assim, neste capítulo são apresentadas técnicas aplicadas a classificadores tradicionais (proposicionais).

2.1 Principais Conceitos

Quando as pessoas têm que tomar uma decisão difícil, geralmente elas preferem levar em consideração opiniões de vários especialistas do que considerar a opinião de um único assessor. Por exemplo, antes de escolher uma nova política em uma organização, um líder ditador pode ser mal aconselhado pela opinião do seu único assessor e tomar uma decisão errada. Em um contexto democrático, uma discussão com diferentes pontos de vista de vários assessores pode levar a um consenso, senão, pode-se usar uma votação com todos os especialistas. Assim, pode-se combinar opiniões de diferentes assessores.

Em aprendizado de máquina técnicas de *ensemble* procuram modelar a estratégia de tomada de decisão comentada acima. Nela, um modelo gerado por um algoritmo de aprendizado pode ser representado como um especialista

(dependendo da qualidade dos dados de treino e se o algoritmo é apropriado para o problema analisado). Uma abordagem óbvia para obter decisões mais confiáveis é combinar a saída de diferentes modelos.

Um classificador *ensemble* é um conjunto de classificadores cujas predições individuais são combinadas de alguma forma (tipicamente por votação) para classificar novos exemplos. Técnicas de *ensemble* constituem uma área ativa de pesquisa em aprendizado supervisionado na qual se estuda métodos para construção de melhores classificadores. Em geral, classificadores *ensembles* são mais precisos do que os classificadores individuais que os originaram (Dietterich, 2000a).

As condições suficientes e necessárias para que um *ensemble* seja mais preciso que seus modelos individuais é que seus classificadores sejam precisos e diversos (Hansen e Salamon, 1990a). Um classificador é preciso se tem uma taxa de erro melhor que um classificador aleatório para novos exemplos. Dois classificadores são diversos se eles cometem erros diferentes em novos exemplos. Para entender melhor porque acurácia e diversidade são fatores importantes, imagine-se que se tem um *ensemble* de três classificadores $\{C_1, C_2, C_3\}$ e um novo exemplo x. Se os 3 classificadores são idênticos (ou seja, não diversos), então quando $C_1(x_i)$ está errado, $C_2(x_i)$ e $C_3(x_i)$ também estão errados. No entanto, se os erros dos classificadores são não correlacionados, então quando $C_1(x_i)$ está errado, $C_2(x_i)$ e $C_3(x_i)$ podem estar corretos, nesse caso uma votação classificaria corretamente o novo exemplo x_i. Por outro lado, se os classificadores individuais têm taxas de erro não correlacionadas excedendo 0,5, então a taxa de erro do classificador *ensemble* aumentará como resultado da votação. Por isso, é bom criar um classificador *ensemble* usando classificadores individuais com taxa de erro menor do que 0,5 e cujos erros sejam não correlacionados (Dietterich, 2000a).

Geralmente é possível construir bons *ensembles*, devido a existência de três razões fundamentais comentadas a seguir.

1. Estatística: um algoritmo de aprendizado pode ser visto como a busca em um espaço de hipóteses H para identificar a melhor hipótese h nesse espaço. Construindo um *ensemble* usando classificadores com boa acurácia, o algoritmo pode calcular a média dos votos dos modelos e reduzir o risco de escolher uma hipótese ruim. A Figura 2.1(a) ilustra esta situação. A borda exterior denota o espaço de hipóteses H. A borda interior denota o conjunto de hipóteses que tem boa acurácia no conjunto de treino. Sendo o ponto marcado f a hipótese verdadeira, a média das

hipóteses h_1, h_2, h_3 e h_4 seria uma boa aproximação para f e, inclusive, melhor que algumas isoladamente.

2. Computacional: muitos algoritmos de aprendizado trabalham efetuando alguma forma de busca local que pode levar para um ótimo local. Por exemplo, redes neurais, que usam gradiente descendente para minimizar uma função de erro no conjunto de treino, e árvores de decisão, que utilizam uma regra de divisão gulosa para produzir a árvore de decisão. Assim, é computacionalmente difícil encontrar a melhor hipótese para o algoritmo de aprendizado. No entanto, um classificador *ensemble* usando uma busca local que parte de diferentes pontos pode produzir uma melhor aproximação para a hipótese verdadeira desconhecida, como é apresentado na Figura 2.1(b).

3. Representacional: na maioria de aplicações de aprendizado de máquina, eventualmente a função verdadeira f não pode ser representada por nenhuma das hipóteses em H. Porém, combinando algumas hipóteses em H pode-se expandir o espaço H e encontrar uma boa aproximação para f, como é apresentado na Figura 2.1(c).

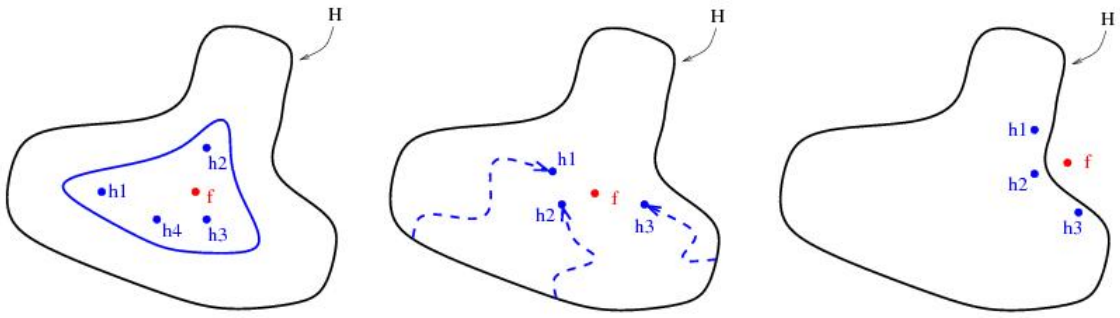


Figura 2.1: Razões fundamentais para criar *ensembles*: (a) razão estatística. (b) razão computacional. (c) razão representacional. Fonte (Dietterich, 2000a).

2.2 Métodos para construir *ensembles*

De acordo com Dietterich (2000a) há diferentes métodos para construir *ensembles*. Nesta seção, alguns métodos de acordo com a classificação de Dietterich (2000a) são apresentados.

2.2.1 Votação Bayesiana

Na votação Bayesiana, cada modelo h define uma distribuição de probabilidade condicional: $h(x_i) = P(y_i = c_k|x_i, h)$, onde a predição da classe c_k depende do novo exemplo x_i a classificar e do modelo h induzido. Considerando um novo exemplo x_i e um conjunto de treino E_{tr} o problema de classificação pode ser apresentado como $P(y_i = c_k|E_{tr}, x_i)$. Podemos reescrever esta equação como uma soma ponderada entre todas as hipóteses de H, como é apresentado na Equação 2.1.

$$P(y_i = c_k|E_{tr}, x_i) = \sum_{h \in H} h(x_i)P(h|E_{tr}) \tag{2.1}$$

Pode-se ver este classificador como um *ensemble*. Esse *ensemble* consiste em todas as hipóteses $h \in H$ ponderadas pela sua probabilidade posterior $P(h|E_{tr})$.

2.2.2 Manipulação de exemplos

A abordagem por manipulação de exemplos considera o conjunto dos exemplos de treino para gerar diferentes classificadores (hipóteses). O algoritmo de aprendizado é executado várias vezes com diferentes subconjuntos dos exemplos de treino. Este método funciona sobretudo com algoritmos instáveis (algoritmos cuja saída do classificador pode mudar muito com pequenas mudanças no conjunto de treino), como as árvores de decisão.

Nesse contexto, uma organização apresentada em Rokach (2010) classifica métodos *ensembles* em duas categorias de acordo com em que medida cada classificador afeta a criação dos proximos classificadores. Assim, tem-se os classificadores *dependentes* e os *independentes*. Na categoria dependente, o resultado de um determinado classificador afeta a criação do classificador seguinte, já na categoria independente, cada classificador é construído independentemente e seus resultados são combinados de alguma forma. A seguir, estas categorias são detalhadas.

1. Métodos Dependentes: os métodos dependentes apresentam uma interação entre os classificadores do *ensemble*. Portanto, é possível aproveitar o conhecimento gerado em iterações anteriores para orientar a aprendizagem nas próximas iterações. Este processo, ilustrado na Figura 2.2, consiste nas etapas de treino e classificação. Primeiramente, a etapa de treino considera os seguintes pasos: (i) seleção de um conjunto de

exemplos parar criar o conjunto de treino E_{tri}; (ii) indução do modelo h_i aplicando o classificador L ao conjunto de exemplos E_{tri}; (iii) seleção dos dados de treino no conjunto $E_{tr(i+1)}$ considerando o modelo h_i e o conjunto de treino inicial E_{tr}; (iv) armazenamento dos modelos induzidos ($h_1, h_2, ..., h_T$). Posteriormente, a etapa de classificação combina todos os modelos induzidos em um classificador final (*Boosting*), ou usa apenas o último classificador produzido (Incremental Batch Learning) (Rokach, 2010).

- *Boosting*: um algoritmo conhecido dessa abordagem é o *Adaboost* (Freund e Schapire, 1996, 1995; Schapire, 1997; Schapire et al., 1998). Este algoritmo considera os exemplos de treino para gerar diferentes classificadores (hipóteses) atribuindo um peso para cada exemplo. Os exemplos mais difíceis de classificar terão pesos maiores e os mais fáceis pesos menores nas iterações seguintes.

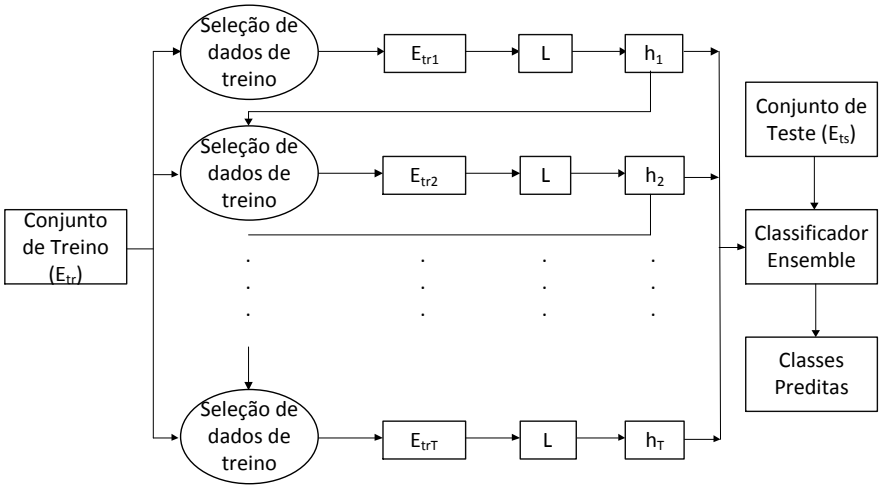


Figura 2.2: Fases no processo de métodos dependentes.

- *Incremental Batch Learning* (Rokach, 2010): neste método a classificação produzida em uma iteração é dada como conhecimento prévio ao algoritmo de aprendizado na seguinte iteração. Conforme a Figura 2.3, o algoritmo de aprendizado utiliza o conjunto de treino atual E_{tri} com o modelo atual h_i para criar o conjunto E_{tr_i+1}, e aplicando o classificador L é gerado o modelo h_{i+1}. Porém, apenas o modelo construído na última iteração (h_T) é utilizado como classificador final.

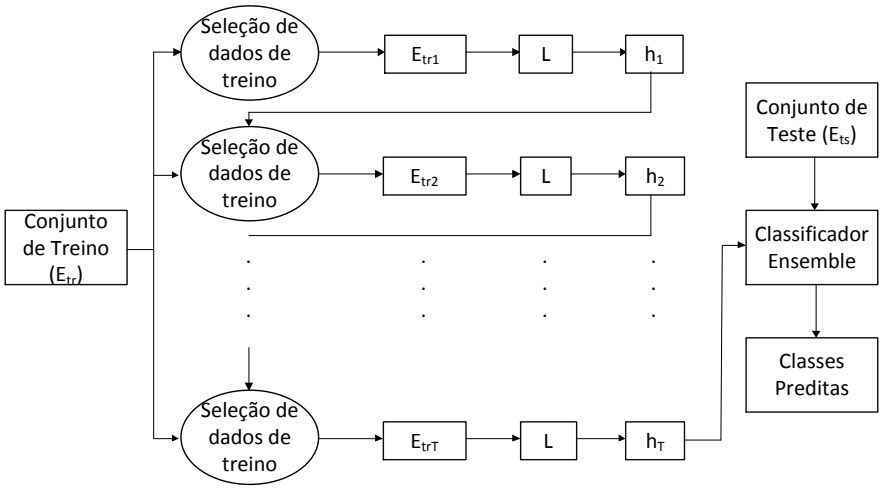


Figura 2.3: Fases no processo do *Incremental Batch Learning*.

2. Métodos Independentes: nestes métodos o conjunto de dados original é dividido em vários subgrupos a partir dos quais vários classificadores são induzidos. Estes subgrupos podem ser disjuntos ou com sobreposição. A seguir, algum processo de combinação dos classificadores é aplicado para gerar o classificador final. Este comportamento é ilustrado na Figura 2.4, mantendo as etapas da Figura 2.2 de métodos dependentes, porém a etapa de criação do conjunto de treino $E_{tr(i+1)}$ é gerada só considerando o conjunto de treino inicial E_{tr}, sem considerar o modelo h_i.

   Algumas técnicas desta abordagem são o *Bagging*, o *Wagging* e os *Cross validated committees* apresentadas a seguir.

   - *Bagging* (Breiman, 1996): em cada execução apresenta-se para o algoritmo de aprendizado n elementos do conjunto de treino selecionados aleatoriamente com reposição dos n exemplos do treino.
   - *Wagging* (Bauer e Kohavi, 1999): este método é uma variante do *Bagging*, onde cada classificação é induzida usando todo o conjunto de treino, porém cada instância tem um peso atribuído.
   - *Cross validated committees* (Parmanto et al., 1995): esta abordagem constrói conjuntos de treino desconsiderando alguns subconjuntos dos exemplos de treino. Por exemplo, o conjunto de treino pode ser dividido em T subconjuntos disjuntos. A seguir, pode-se construir T conjuntos de treino diferentes desconsiderando um subconjunto destes T em cada iteração. Em seguida, para cada con-

junto de treino é aplicado um classificador L e criado um modelo de classificação h_i. Finalmente, para classificar novos exemplos os modelos de classificação $(h_1, h_2, ..., h_T)$ são combinados.

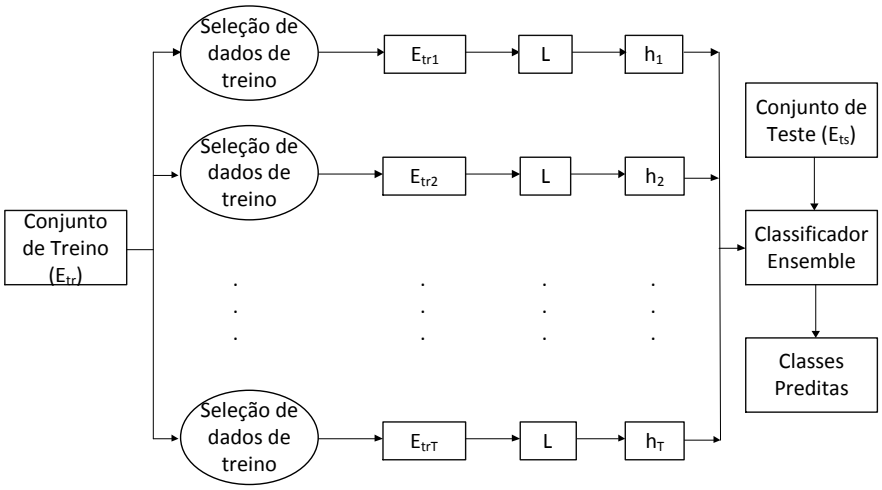


Figura 2.4: Fases no processo de métodos independentes.

2.2.3 Manipulação de vetores de atributos

A manipulação de vetores de atributos subdivide o vetor de atributos disponível para o algoritmo e cria um modelo de classificação para cada subconjunto dos atributos. A seguir, na etapa de classificação os modelos são combinados. Por exemplo, no trabalho de Cherkauer (1996) utiliza-se um *ensemble* de 32 redes neurais para identificar vulcões em Vênus. Estas redes consideraram 8 diferentes subconjuntos de 119 características e 4 tamanhos diferentes nas redes. Estes subconjuntos foram selecionados considerando algumas operações de processamento de imagens, como, *principal componente analysis* e *fast fourier transform*. Do mesmo modo, Tumer e Ghosh (1996) usaram uma técnica similar para um conjunto de dados de navegação e mapeamento pelo som com um vetor de características de tamanho 25. Evidentemente, esta abordagem é executada com êxito quando as características são muito redundantes (Cherkauer, 1996; Tumer e Ghosh, 1996).

2.2.4 Manipulação das classes de saída

A abordagem de manipulação de classes considera os valores das classes de saída ou rótulos do algoritmo de aprendizado. Suponha que o problema de

classificação tenha um número k de classes. São gerados dois subconjuntos A_i e B_i distribuindo aleatoriamente as k classes nestes subconjuntos. Em seguida, as classes originais no conjunto A_i são re-rotuladas com 0 e no B_i com 1. Este novo conjunto de treino re-rotulado é a entrada para o algoritmo de aprendizado que constrói um classificador binário h_i. Repetindo este processo T vezes (gerando diferentes subconjuntos A_i e B_i), temos um *ensemble* de T classificadores binários $\{h_1, h_2, ..., h_T\}$.

Para classificar um novo exemplo x_j, cada hipótese h_i classificará x_j. Se $h_i(x_j) = 0$ cada classe de A_i recebe um voto e se $h_i(x_j) = 1$ cada classe de B_i recebe um voto. Depois que os T classificadores votarem, a classe com o maior número de votos é a classe predita.

Dietterich e Bakiri (1995) discutem que esta abordagem melhora o desempenho da árvore de decisão C4.5 e da rede neural *backpropagation* com um conjunto de problemas difíceis de classificar usando o algoritmo *Error-correcting output coding* (apresentado na seção 2.3.4). Além disso, Schapire et al. (1998) combinaram o *Adaboost* com o algoritmo *Error-correcting output coding* criando um melhor algoritmo *ensemble* de classificação *AdaBoost OC* (Schapire et al., 1998). Finalmente, Ricci e Aha (1997) usaram o algoritmo de *Error-correcting output coding* com uma técnica de seleção de características.

2.3 Técnicas

Nesta seção, são detalhadas as principais técnicas para algoritmos *ensembles*.

2.3.1 *Bagging*

A técnica de *Bagging* foi proposta por Breiman (1996) e considera vários subconjuntos de treino (amostras) do mesmo tamanho selecionados aleatoriamente do domínio do problema analisado com reposição. Devido a reposição, alguns exemplos podem ser escolhidos mais de uma vez e outros desconsiderados. Para cada um desses subconjuntos é usado como classificador base L um algoritmo de aprendizado como uma árvore de decisão, uma rede neural, entre outros, gerando um modelo de classificação h. Os modelos de classificação ($h_1, h_2, ..., h_T$) gerados têm que cumprir a seguinte propriedade: "pequenas mudanças no conjunto de treino, grandes mudanças no resultado da classificação". Isto implica que para novos exemplos alguns classificadores geram a classe correta, mas outros não. Este processo é resumido no

Algoritmo 2.1 adaptado de (Witten e Frank, 2005).

Entrada:

- E_{tr}: conjunto de exemplos de treino.
- L: classificador base.
- T: número de iterações.

Saída:

- $h_1, h_2, ..., h_T$: T modelos de classificação.

Seja n o número de exemplos no conjunto de treino E_{tr}.
para todo $t \leftarrow 1$ *até* T **faça**
  $E_{tr_t} \leftarrow$Selecionar n exemplos com reposição do conjunto de treino E_{tr}.
  $h_t \leftarrow$Aplicar o algoritmo de aprendizado L nos exemplos E_{tr_t}.
  Arquivar o modelo gerado h_t.
fim
retorna $h_1, h_2, ..., h_T$

Algoritmo 2.1: Algoritmo *Bagging* - Etapa de Aprendizagem.

A seguir, considerando a analogia com o conjunto de opiniões de vários especialistas. Pode-se atribuir um voto para cada classe determinada pelo modelo gerado no passo anterior. Este processo é apresentado no Algoritmo 2.2 também adaptado de (Witten e Frank, 2005). O processo completo da técnica de *Bagging* é o mesmo dos métodos independentes, ilustrado na Figura 2.4.

Entrada:

- $h_1, h_2, ..., h_T$: modelos de classificação gerados no Algoritmo 2.1.
- T: número de classificadores gerados.
- e: exemplo a classificar, onde $e \in E_{ts}$.

Saída:

- c: classe predita para o exemplo e.

para todo $t \leftarrow 1$ *até* T **faça**
  Predizer a classe do exemplo e usando o modelo h_t.
fim
$c \leftarrow$ determinar a classe que foi predita mais vezes.
retorna c

Algoritmo 2.2: Algoritmo *Bagging* - Etapa de Classificação.

Uma variação do *Bagging* é o *Wagging*, apresentado na seção de métodos independentes. Outra estratégia considerando medidas de diversidade e o *Bagging* é o *Bagging* Usando Diversidade (BUD) (Tang et al., 2006). O algoritmo BUD considera que quanto mais diversos são os classificadores individuais, melhor será o resultado na sua combinação. O algoritmo começa gerando um conjunto de classificadores bases. A seguir, um subconjunto destes classificadores é selecionado considerando medidas de diversidade. Finalmente, o conjunto selecionado é combinado em um classificador final.

Além disso, considerando o método de manipulação de atributos e o *Bagging*, Bryll (2003) propôs o *Attribute Bagging* (AB) que combina conjuntos aleatórios de atributos. Em primeiro lugar, AB procura um tamanho apropriado para os subconjuntos de características. A seguir, seleciona aleatoriamente conjuntos de características e cria projeções usando o conjunto de treino. Em seguida, os classificadores são induzidos em cada projeção. Finalmente, os classificadores são combinados por votação.

Adicionalmente, em um estudo mais recente foi desenvolvido o algoritmo *Trimmed Bagging* (TB) (Croux et al., 2007). O objetivo deste algoritmo é podar classificadores com taxas de erro elevadas. Avaliações experimentais mostraram que o TB apresentou resultados comparáveis com o *Bagging* em classificadores instáveis. Porém, apresentou melhores resultados em classificadores estáveis, devido ao fato de utilizar classificadores com taxas de erro baixas.

2.3.2 *Boosting*

Para introduzir o conceito de *Boosting* considere a seguinte situação. Uma pessoa que gosta muito de corridas de cavalos aposta em todos os cavalos: para os cavalos que ganham mais corridas aposta mais, e para os outros, ele aposta menos. Por outro lado, ela agora deseja encontrar uma maneira de melhorar suas apostas, então decide conversar com um apostador experiente perguntando-lhe sobre suas estratégias. Para o apostador é difícil combinar todas suas estratégias, porém para determinadas corridas existem algumas (por exemplo, aposte no cavalo com mais vitórias, aposte no cavalo melhor treinado, entre outras). Agora, a pessoa tem um conjunto de muitas estratégias, no entanto tem alguns problemas:

- como escolher o conjunto de corridas apresentadas ao apostador experiente para extrair as estratégias que sejam mais úteis?

- Como unir as estratégias em uma só que melhore a predição dos resultados, em relação ao uso separado das estratégias?

A técnica de *Boosting* aborda o problema geral de produzir uma estratégia mais adequada combinando estratégias pouco precisas. Fazendo uma definição mais formal, o *Boosting* produz um classificador com alta acurácia combinando vários classificadores "fracos", de modo que as acurácias destes classificadores sejam melhoradas (Meir e Rätch, 2003). Este processo é apresentado na Equação 2.2, onde α_t representa um peso ponderado do classificador e $h_t(x_i)$, um dos classificadores a combinar.

$$y_i = \sum_{t=1}^{T} \alpha_t . h_t(x_i) \tag{2.2}$$

Uma primeira origem desta técnica foi no modelo de aprendizado PAC (*Probably Approximately Correct*) (Valiant, 1984). Kearns e Valiant (1988, 1994) foram os primeiros a perguntar se um algoritmo de classificação "fraco" no modelo PAC podia se tornar um classificador com boa acurácia. Ao mesmo tempo, Schapire (1989) desenvolveu o primeiro algoritmo de *Boosting* em tempo polinomial. A seguir, Freund (1990) desenvolveu um algoritmo mais eficiente de *Boosting*. Os primeiros experimentos com estes algoritmos foram testados em um trabalho de OCR (*Optical Character Recognition*) por Drucker et al. (1993).

A técnica de *Boosting* é iterativa, e cada novo modelo gerado é influenciado pelo desempenho dos modelos gerados anteriormente. O objetivo do *Boosting* é criar novos modelos que classifiquem bem os exemplos mal classificados pelos modelos antigos, portanto, sua estratégia é concentrar-se nos exemplos mal classificados. Por exemplo, assumindo-se que há 8 exemplos de treino e que o exemplo 1 é de classificação difícil. Então, em cada conjunto de treino ao longo das iterações do algoritmo, o exemplo 1 será apresentado mais vezes ao indutor (conforme as iterações da Tabela 2.1).

Conjunto de treino	exemplos
Conjunto de treino (Original):	1, 2, 3, 4, 5, 6, 7, 8
Conjunto de treino 1 (*Boosting*):	2, 7, 8, 3, 7, 6, 3, 1
Conjunto de treino 2 (*Boosting*):	1, 4, 5, 4, 1, 5, 6, 4
Conjunto de treino 3 (*Boosting*):	7, 1, 5, 8, 1, 8, 1, 4
Conjunto de treino 4 (*Boosting*):	1, 1, 6, 1, 1, 3, 1, 5

Tabela 2.1: Exemplo *Boosting*.

Além disso, essa técnica dá um peso α_t (conforme a Equação 2.2) a cada um dos modelos gerados de acordo com o quão bem eles classificam os dados de treino. Assim, modelos com erro pequeno terão um peso maior na classificação de novos exemplos, e os com erro grande terão um peso menor.

Já em um exemplo visual, considerando o conjunto de dados apresentado na Figura 2.5(a), os exemplos mal classificados pelo primeiro classificador h_1 aumentarão seu peso na próxima distribuição (representado pelo tamanho do exemplo), conforme a Figura 2.5(b). Logo, um novo classificador h_2 é gerado considerando os pesos da nova distribuição, conforme a Figura 2.5(c), atribuindo maior peso aos exemplos difíceis de classificar. Finalmente, na Figura 2.5(d) é apresentado o classificador final da combinação dos modelos (h_1, h_2, h_3 e h_4) gerados considerando seus pesos α_1, α_2, α_3 e α_4.

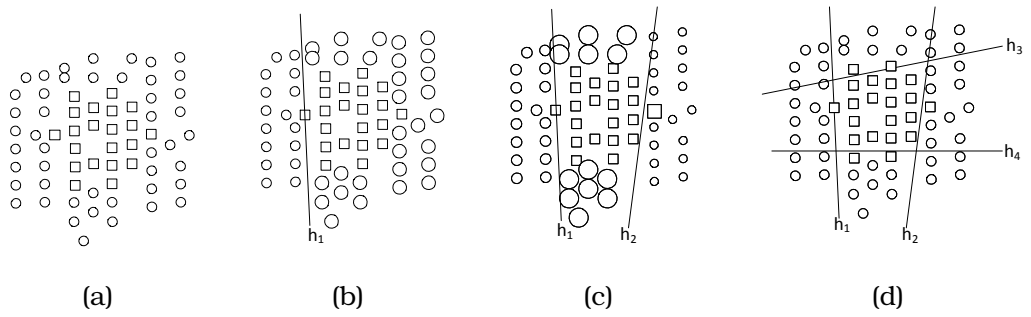


Figura 2.5: Exemplo de execução da técnica de *Boosting*: (a) conjunto de dados inicial, (b) conjunto de dados classificados com o classificador "fraco" (Primeira execução), (c) conjunto de dados classificados com o classificador "fraco" (Segunda execução) e (d) construção do classificador final combinando os classificadores "fracos". O tamanho do exemplo representa seu peso.

Um algoritmo muito utilizado é o *Adaboost*, desenvolvido por Freund e Schapire (1995) e apresentado nos Algoritmos 2.3 e 2.4 (adaptados de Witten e Frank (2005)). O algoritmo tem como entrada n exemplos de treino, um classificador L e o número de iterações T. No começo, o algoritmo dá pesos iguais aos exemplos de treino ($D_1(i) = 1/n$). Em seguida, o classificador é treinado e testado ainda com os exemplos de treino. Os exemplos de treino mal classificados terão o seu peso D_i aumentado, enquanto que os exemplos bem classificados terão o seu peso diminuído. Além disso, é verificado o erro no conjunto de treino de cada classificador. Se o erro ultrapassar 0,5, então a técnica elimina o classificador correspondente e termina o algoritmo. O mesmo processo ocorre quando o erro é 0. Depois de executar as T iterações, o algoritmo terá como saída os modelos de classificação $(h_1, h_2, ..., h_T)$ e os pe-

sos $(\beta_1, \beta_2, ..., \beta_T)$, onde $\beta_i = (1 - e_i)/e_i$ e e_i é o erro do modelo na iteração i. A classificação de novos exemplos é feita por votação com peso, sendo que cada classificador contribui com $\alpha_i = ln(\beta_i)$ votos, sendo ln o logaritmo neperiano. Este critério faz com que os classificadores com menores taxas de erro tenham pesos mais altos e vice-versa.

Entrada:

- E_{tr}: conjunto de exemplos de treino.
- L: classificador.
- T: número de iterações.

Saída:

- $h_1, h_2, ..., h_T$: T modelos gerados.
- $\beta_1, \beta_2, ..., \beta_T$: pesos dos T modelos gerados.

Seja n o número de exemplos no conjunto de treino E_{tr}.
$D_1(i) \leftarrow 1/n$, para cada um dos n exemplos.
para todo $t \leftarrow 1$ *até* T **faça**
  $h_t \leftarrow$ aplicar L aos exemplos usando a distribuição D_t.
  $e_t \leftarrow$ soma das distribuições $D_t(i)$ para os exemplos mal classificados com h_t.
  se $e_t \geq 0,5$ *ou* $e_t == 0$ **então**
    Fim do algoritmo.
  fim
  $\beta_t \leftarrow (1 - e_t)/e_t$.
  para todo $i \leftarrow 1$ *até* n **faça**
    se h_t *classifica mal o exemplo* i **então**
      $D_{t+1}(i) = D_t(i) * \beta_t$.
    senão
      $D_{t+1}(i) = D_t(i)$.
    fim
  fim
  Normalizar D_{t+1} tal que a soma seja 1.
fim
retorna $h_1, h_2, ..., h_T$ *e* $\beta_1, \beta_2, ..., \beta_T$

Algoritmo 2.3: Algoritmo *Adaboost* - Etapa de Aprendizagem.

2.3.3 *Stacking*

A técnica de *Stacking* ou *Stacked Generalization* foi desenvolvida por Wolpert (1992). Esta técnica trabalha de maneira diferente ao combinar classifica-

Entrada:

- $h_1, h_2, ..., h_T$: classificadores gerados no Algoritmo 2.3.
- $\beta_1, \beta_2, ..., \beta_T$: pesos gerados no Algoritmo 2.3.
- T: número de modelos gerados.
- e: exemplo a classificar, onde $e \in E_{ts}$.

Saída:

- c: classe estimada do exemplo e.

atribuir o peso 0 para todas as classes.
para todo $t \leftarrow 1$ *até* T **faça**
  $\alpha_t = ln(\beta_t)$.
  Adicionar α_t ao peso da classe predita pelo modelo $h_t(e)$.
fim
$c \leftarrow$ determinar a classe com maior peso.
retorna c

Algoritmo 2.4: Algoritmo *Adaboost* - Etapa de Classificação.

dores. Enquanto as técnicas de *Bagging* e *Boosting* utilizam só um algoritmo de classificação, o *stacking* considera vários algoritmos de classificação na sua construção, caracterizando-se como um *ensemble* heterogêneo.

Suponha que existem T classificadores diferentes $\{C_1, C_2, ..., C_T\}$. Cada classificador é treinado com o conjunto de treino para predizer uma classe c_t para cada exemplo. Então, ter-se-á um conjunto de classes preditas $\{c_1, c_2, ..., c_T\}$ usando os classificadores do passo anterior. Este conjunto de classes é a entrada para um novo classificador M_c (meta-classificador) que predirá a classe final, como é apresentado no Algoritmo 2.5.

Para classificar um novo exemplo e cada modelo h_t gerado predirá sua classe c_t. Estas classes são armazenadas em um vetor $v = (c_1, c_2, ..., c_T)$. Com este vetor v e o meta-classificador M_c será determinada a classe final. Este processo é apresentado no Algoritmo 2.6.

2.3.4 *Error Correcting Output Code*

A técnica *Error Correcting Output Code* foi proposta por Dietterich e Bakiri (1995) e procura melhorar o desempenho de classificadores em problemas de classificação multi-classe. É possível transformar um problema multi-classe em um problema de duas classes. Para cada classe, um conjunto de dados é gerado contendo uma cópia dos exemplos do conjunto original, porém

Entrada:

- E_{tr}: conjunto de exemplos de treino.
- T: número de classificadores.
- $(C_1, C_2, ..., C_T)$: classificadores bases.

Saída:

- M_c: classificador final (meta-classificador).
- $(h_1, h_2, ..., h_T)$: modelos de classificação.

$(h_1, h_2, ..., h_T) \leftarrow$Treinar T classificadores $(C_1, C_2, ..., C_T)$ com o conjunto de treino E_{tr}.
$(c_1, c_2, ..., c_T) \leftarrow$Obter as classes preditas dos modelos $(h_1, h_2, ..., h_T)$ no conjunto de treino E_{tr}.
$v \leftarrow (c_1, c_2, ..., c_T)$.
Treinar um novo classificador M_c com o vetor de características v.
retorna M_c **e** $(h_1, h_2, ..., h_T)$

Algoritmo 2.5: Algoritmo *Stacking* - Etapa de Aprendizagem.

Entrada:

- T: número de classificadores.
- e: exemplo a classificar, onde $e \in E_{ts}$.
- $h_1, h_2, ..., h_T$: modelos de classificação gerados no Algoritmo 2.5.
- M_c: meta-classificador.

Saída:

- c: classe predita para o exemplo e.

para todo $t \leftarrow 1$ *até* T **faça**
| $c_t \leftarrow$ Predizer a classe do exemplo e usando o modelo h_t.
fim
$v \leftarrow (c_1, c_2, ..., c_T)$.
$c \leftarrow$ determinar a classe final usando o classificador M_c e o vetor de características v.
retorna c

Algoritmo 2.6: Algoritmo *Stacking* - Etapa de Classificação.

com o valor da classe modificada. Se o exemplo tem a classe em questão ela é associada com a classe 1, caso contrário, com a classe 0. A seguir, na etapa de treino são construídos classificadores para cada um destes conjuntos binários. Posteriormente, na etapa de classificação, o novo exemplo é testado com os classificadores binários, e a classe final é dada pelo classificador com maior confiança.

Por exemplo, considerando um problema com 4 classes (a, b, c e d) cada classe pode ser convertida em um código de tamanho 4 (*4-bits*). Posteriormente, na etapa de classificação gera-se 4 classificadores que predizem cada um destes *bits* de maneira independente (conforme a Tabela 2.2). Por exemplo, se o vetor $v = 0110$ é gerado, são calculadas as distâncias para as classes a,b, c e d, sendo 3, 1, 1 e 3 respectivamente. Assim, as classes b e c apresentam a mínima distância e a classe final é determinada pela classe com maior confiança. Por outro lado, é bom ressaltar que qualquer vetor de quatro *bits* predito diferente dos da Tabela 2.2 teria a mesma distância mínima pelo menos para dois vetores desta tabela. Tal fato atrapalha a predição.

Classe	Vetor de Classe
a	1000
b	0100
c	0010
d	0001

Tabela 2.2: Transformação de um problema multi-classe em um problema de duas classes (método tradicional).

No entanto, o classificador *Error Correcting Output Code* pode corrigir esta situação. Primeiramente, é possível gerar mais classificadores que não só consideram uma classe para atribuir o rótulo 1, mas um subconjunto de classes. Por exemplo, o 3º *bit* na coluna 2 da Tabela 2.2: 0010 atribui o rótulo 1 somente para a classe c, porém, na Tabela 2.3 o 3º *bit* na coluna 2 está representada por 1010, atribuindo o rótulo 1 às classes a e c. Então, em cada iteração um subconjunto de classes terá o novo rótulo 1 e as outras o rótulo 0. A seguir, na etapa de treino são construídos classificadores binários para cada um dos subconjuntos de classes. Formalizando esta etapa, dado um problema de classificação com k classes, cada vetor contém $2^{k-1} - 1$ *bits* e são construídos da seguinte maneira: a primeira classe contém só *bits* um, a segunda classe tem 2^{k-2} *bits* zero seguidos de $2^{k-2} - 1$ *bits* um, a terceira classe tem 2^{k-3} *bits* zero seguidos de 2^{k-3} *bits* um seguidos de 2^{k-3} *bits* zero seguidos de $2^{k-3} - 1$ *bits* um, e assim por diante. O vetor da classe i consiste da alternância de

conjuntos de 2^{k-i} *bits* zero e um, sendo o último conjunto de tamanho $2^{k-i}-1$. Posteriormente, na etapa de classificação, cada modelo atribuirá um valor 1 ou 0, sendo estes valores combinados em um vetor.

Por exemplo, considerando o problema de 4 classes foi utilizado um vetor de 7 i.é ($2^{4-1}-1$) *bits*. Assim, na Tabela 2.3, a classe a contém só *bits* um, a classe b tem 4(2^{4-2}) *bits* zero seguidos de 3($2^{4-2}-1$) *bits* um, a classe c tem 2(2^{4-3}) *bits* zero seguidos de 2(2^{4-3}) *bits* um seguidos de 2(2^{4-3}) *bits* zero seguido de 1($2^{4-3}-1$) *bit* um, e assim por diante. Já definida a representação das classes, para cada um dos sete bits é treinado um classificador. Posteriormente, na etapa de classificação, utilizando os sete classificadores gerados é criado, por exemplo, o vetor $v = 1011111$. Em seguida, procura-se a classe do vetor mais próximo para v, na Tabela 2.3. Este é o vetor da classe a, porque só é diferente em um *bit* apenas, então o novo exemplo é da classe a. Para encontrar a classe do vetor final é utilizada a distância de *Hamming*, que mede a discrepância entre os códigos de *bits*. As distâncias para as classes a, b, c e d são 1, 3, 3 e 5 respectivamente. Então, pode-se concluir que o classificador teve um erro na classificação e identifica a classe a como sua classe verdadeira.

Classe	Vetor de Classe
a	1111111
b	0000111
c	0011001
d	0101010

Tabela 2.3: Transformação de um problema multi-classe em um problema de duas classes (método *error-correcting code*).

Finalmente, considerando os exemplos de quatro e sete *bits*, a pergunta é como determinar se é possível realizar uma correção de erro? O número de erros que podem ser corrigidos é calculado usando a equação $(d-1)/2$, onde d representa a distância mínima entre qualquer par de vetores de *bits*. O valor de d no caso de *4-bits* é 2, conforme a Tabela 2.2, podendo corrigir 0 erros. No caso de *7-bits* o valor de d é 4, conforme a Tabela 2.3, podendo corrigir um erro nos *bits*. Maiores detalhes desse algoritmo podem ser encontrados em Witten e Frank (2005).

CAPÍTULO

3

Classificadores Relacionais

Neste capítulo, são abordados classificadores relacionais baseados em grafos considerando seus principais conceitos e algoritmos. Estes algoritmos de classificação serão utilizados posteriormente como classificadores base para as abordagens de *ensemble* relacional propostas.

Em geral, os classificadores baseados em grafos consideram três componentes: (1) um modelo relacional, para classificar vértices não rotulados considerando os *links* para seus vizinhos; (2) um procedimento de inferência coletiva, para aplicar um processo de inferência em vértices interligados não rotulados, ilustrado na Figura 3.1 (os vértices não rotulados são representados pelo símbolo "?"), e (3) um modelo não relacional, o qual utiliza informação local (quando disponível) representada por uma tabela atributo-valor na classificação. Os modelos não relacionais geram probabilidades a priori para o aprendizado relacional e o processo de inferência coletiva.

Nas seguintes seções, são apresentados o classificador de hipertexto (Chakrabarti et al., 1998) e o classificador baseado em *links* (Lu e Getoor, 2003), os quais utilizam informações individuais (representadas por uma tabela atributo-valor) e relacionais (representadas por grafos). Posteriormente, são apresentados algoritmos baseados somente em informações relacionais, propostos por (Chakrabarti et al., 1998; Lu e Getoor, 2003; Macskassy e Provost, 2007).

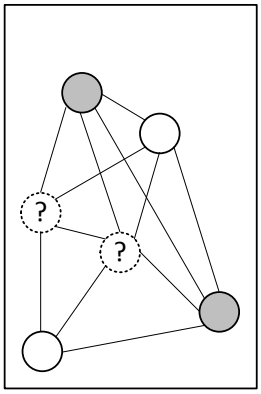


Figura 3.1: Exemplos de treino e teste em um grafo. O conjunto de treino é representado pelos vértices cinza e branco. Por outro lado, os vértices de teste são representados pelo símbolo "?".

3.1 Classificadores baseados em informações individuais e relacionais

Nesta seção, são apresentados algoritmos de classificação baseados em informações individuais e relacionais.

- Classificador de Hipertexto: o classificador de Hipertexto, proposto por Chakrabarti et al. (1998), utiliza informações relacionais além das individuais para classificação de documentos (páginas *Web*, patentes, etc), explorando novas maneiras de classificar informação baseada em *hyperlinks* . A classificação é feita utilizando um grafo G, onde os vértices representam um documento a classificar x_i em uma das possíveis classes c_k. Este documento tem um texto associado t_i e uma vizinhança N_i de documentos associados com ele. A classificação é baseada na probabilidade máxima de x_i pertencer a uma classe c_k, dada pela Equação 3.1.

$$P(y_i = c_k|t_i, N_i) = P(y_i = c_k|t_i).P(y_i = c_k|N_i) \tag{3.1}$$

  Algumas adaptações desta estratégia são:

  1. *Local*: somente utiliza a informação textual, desconsiderando a informação de *hyperlinks*.

2. *Local+Nbr*: somente considera a informação textual, porém o texto completo de todos os vizinhos são concatenados com o texto do documento.
3. *Local+TagNbr*: esta adaptação é semelhante ao *Local+Nbr*, porém mantém separada a informação local do texto fazendo uma diferenciação da informação obtida dos vizinhos. Assim, o classificador pode diferenciar o conteúdo local do conteúdo não local.

Além disso, se todas as classes dos documentos vizinhos do documento x_i são conhecidas, obtém-se uma classificação adequada. Caso contrário, se vizinhos com classe desconhecida estão interligados, deve-se aplicar um critério de inferência coletiva. Finalmente, Chakrabarti et al. (1998) afirmam que este é o primeiro classificador que combina informação individual e relacional para classificação de exemplos textuais.

- Classificador baseado em *links*: o classificador baseado em *links* foi proposto por Lu e Getoor (2003) e utiliza informações individuais dos objetos em conjunto com informações relacionais. As informações individuais e relacionais são representadas por um vetor de características. A seguir, o classificador baseia-se em uma técnica de regressão logística (Hosmer e Lemeshow, 2000) para produzir um modelo de classificação. Em relação às informações relacionais, é gerado um vetor de características usando um dos quatro modelos a seguir (conforme a Figura 3.2).
  - *Mode-link*: utiliza a contagem dos *links* que tem um exemplo em relação a cada classe. Na classe com o maior número de vizinhos é atribuído o valor 1 e para as outras 0.
  - *Count-link*: utiliza a contagem dos *links* que um exemplo possui em relação a cada uma das classes.
  - *Distrib-link*: representa uma versão normalizada do *Count-link*.
  - *Binary-link*: baseia-se no fato que, se um exemplo tem alguma ligação com outro exemplo de uma determinada classe, então nessa classe é atribuído o valor 1. Caso contrário, é atribuído o valor 0.

  Existem ainda problemas quando existem vértices não rotulados interligados. Nesses casos, pode-se empregar o algoritmo de inferência coletiva denominado de classificação iterativa (Lu e Getoor, 2003), proposto pelos próprios autores.

Classificador Baseado em Links			
	○	□	△
Mode-link	1	0	0
Count-link	3	2	0
Distrib-link	0,6	0,4	0
Binary-link	1	1	0

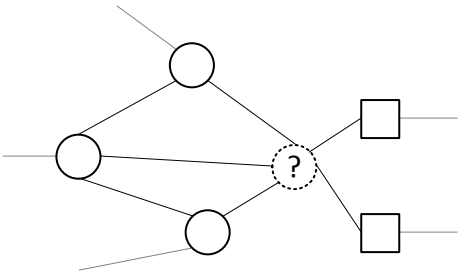


Figura 3.2: Modelos de representação relacional na classificação baseada em *links*. As figuras círculo, quadrado e triângulo representam as classes.

3.2 Classificadores baseados em informações relacionais

A seguir são apresentados classificadores relacionais propostos por Macskassy e Provost (2007) que usam apenas informações relacionais para a classificação.

- Classificador relacional baseado nos vizinhos com votação ponderada (*weighted-vote relational neighbor classifier - wvrn*): o wvrn considera os pesos das arestas do exemplo a classificar. A probabilidade de pertencer a uma classe é definida pela Equação 3.2, na qual x_i é o exemplo a classificar, c_k é uma das possíveis classes dos exemplos, Z é um fator de normalização, N_i são os exemplos ligados ao exemplo x_i, x_j é um vértice adjacente (vizinho) de x_i, $classe(x_j)$ é a classe do vértice x_j e w_{ij} é o peso da aresta entre os exemplos: x_i e x_j.

$$P_1(y_i = c_k|N_i) = \frac{1}{Z} \sum_{(x_j \epsilon N_i | classe(x_j) = c_k)} w_{ij} \tag{3.2}$$

  Uma modificação deste classificador em relação à probabilidade da classificação é dada pela Equação 3.3. Nesta equação é calculada uma

distribuição de probabilidade ($P_1(y_j = c_k|N_j)$) de classes para cada um dos vizinhos de x_i. Este novo classificador foi chamado Classificador probabilístico relacional baseado nos vizinhos (*Probabilistic Relational Neighbor classifier - prn*)

$$P(y_i = c_k|N_i) = \frac{1}{Z} \sum_{(x_j \epsilon N_i | classe(x_j) = c_k)} w_{ij}.P_1(y_j = c_k|N_j) \tag{3.3}$$

- Classificador relacional baseado na distribuição de classes dos vizinhos (*class-distribution relational neighbor classifier - cdrn*): este classificador é baseado em 2 vetores, um vetor de classes (VC) obtido para cada exemplo e um vetor de referências (VR) obtido para cada classe. O vetor de classes VC contém o somatório de pesos de todas as arestas que ligam o exemplo x_i a vértices de uma determinada classe c_k (conforme a Equação 3.4), e o vetor de referências VR é a normalização dos vetores de classe VC de todos os exemplos que são da classe c_k (conforme a Equação 3.5). Nessas equações, x_i é o vértice que será classificado, N_i a vizinhança do vértice x_i, x_j denota um vizinho de x_i, $classe(x_j)$ é a classe do vértice x_j, w_{ij} é o peso da aresta entre os vértices x_i e x_j e X_{c_k} é o conjunto de exemplos rotulados que são da classe c_k. A classificação final é obtida ao calcular a probabilidade do exemplo x_i pertencer à classe c_k, conforme a Equação 3.6. Nesta equação, a função de similaridade empregada foi cosseno.

$$VC(x_i, c_k) = \sum_{(x_j \epsilon N_i | classe(x_j) = c_k)} w_{ij} \tag{3.4}$$

$$VR(c_k) = \frac{1}{|X_{c_k}|} \sum_{(x_i \epsilon X_{c_k})} VC(x_i, c_k) \tag{3.5}$$

$$P(y_i = c_k|N_i) = similaridade(VC(x_i, c_k), VR(c_k)) \tag{3.6}$$

- Classificador Bayesiano baseado apenas na rede (*Network-only bayes classifier - nbc*): este classificador é baseado no classificador de hipertexto (Chakrabarti et al., 1998), porém desconsidera as informações individuais. O nbc utiliza uma adaptação do classificador bayesiano baseado na vizinhança do vértice que será classificado, dada pela Equação 3.7.

$$P(y_i = c_k|N_i) = P(N_i|c_k).P(c_k) \tag{3.7}$$

A probabilidade $P(c_k)$ é calculada como a proporção dos exemplos da classe c_k nos exemplos rotulados e a probabilidade $P(N_i|c_k)$ é calculada pela Equação 3.8, levando em conta a probabilidade de x_j ser da sua classe verdadeira c_j dado que a classe de x_i é c_k ponderado pelo peso da aresta w_{ij}. Onde, N_i são os vizinhos do exemplo que será classificado x_i, x_j todos os vértices que pertencem a N_i, c_j a classe do exemplo x_j (informação já conhecida pois x_j é um exemplo rotulado), c_k uma das possíveis classes e w_{ij} o peso na aresta entre os vértices x_i e x_j.

$$P(N_i|c_k) = \frac{1}{Z} \prod_{x_j \in N_i} P(classe(x_j) = c_j | y_i = c_k)^{w_{ij}} \tag{3.8}$$

- Classificador baseado apenas nas conexões da rede (*Network-only link based classification - nlb*): este classificador é uma adaptação do classificador baseado em *links* (Lu e Getoor, 2003). No entanto, utiliza somente as informações relacionais. Este método cria um vetor de características para cada vértice considerando as classes dos vértices vizinhos. A seguir, é utilizada uma técnica de regressão logística (Hosmer e Lemeshow, 2000) para induzir o classificador, de uma maneira similar ao classificador baseado em *links*. Para a criação do vetor de características, utiliza-se os modelos relacionais *count-link* (nlb-b), *distrib-link* (nlb-d), *binary-link* (nlb-b) ou *mode-link* (nlb-m). Sendo o *count-link* o que apresentou os melhores resultados.

CAPÍTULO

4

Ensembles na Classificação Relacional

Neste Capítulo, são apresentadas as técnicas desenvolvidas durante o mestrado para exploração e mapeamento da estratégia de *boosting* tradicional em uma estratégia de *boosting* relacional, que deu origem ao algoritmo KNN Adaptativo baseado em Grafos (A-KNN). Também são apresentados os classificadores *ensembles* propostos baseados em grafos.

4.1 Exploração do algoritmo *Boosting* relacional

Nesta subseção, são apresentados os experimentos preliminares na investigação e busca por uma estratégia de *boosting* para classificadores baseados em grafos. Esse histórico é relevante para o entendimento de alguns critérios e considerações adotadas na proposta final da técnica de *boosting* relacional.

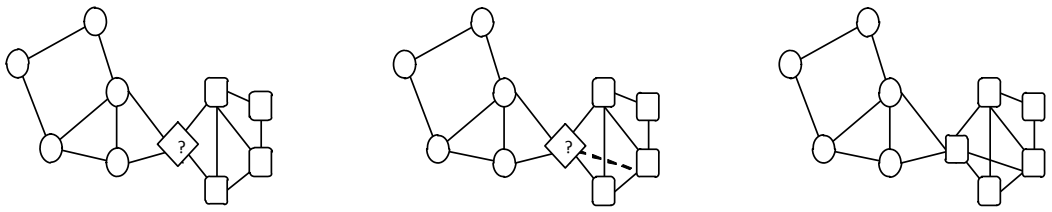
4.1.1 Proposta

A proposta apresentada tem como objetivo principal melhorar o desempenho de classificadores relacionais baseados em grafos. Este problema, na classificação proposicional, é endereçado usando, entre outras técnicas, a técnica de *Boosting*. Assim, procurou-se uma maneira de adaptar esse algoritmo para dados representados em grafos. Uma idéia inicial foi mapear a alteração na distribuição de probabilidade de um exemplo x_i, do método

proposicional, na alteração da distribuição do grau desse vértice x_i. Por outro lado, considerando que os dados originais são proposicionais, isto é, representados por uma tabela atributo-valor, esses dados primeiramente deveriam ser representados em um grafo. Utilizou-se nessa investigação os grafos KNN e sua variante com arestas dirigidas e pesadas. Observa-se, porém, que existem outras possibilidades, como as redes ε, já comentadas.

Na Figura 4.1(a) apresenta-se um vértice difícil de classificar, indicado com o símbolo "?". Este vértice é difícil de classificar, pois está ligado ao mesmo número de vértices da classe círculo e da classe quadrado. A hipótese de pesquisa é que alterando a distribuição dos graus dos vértices (vértices mais difíceis de classificar aumentarão seu grau e vértices mais fáceis de classificar diminuirão seu grau) pode implicar em uma melhoria do classificador, conforme as Figuras 4.1(b) e 4.1(c). O incremento de arestas foi realizado procurando os vértices mais próximos do vértice selecionado e a diminuição foi feita eliminando uma aresta aleatóriamente. Assim, o processo iterativo de alteração dos pesos dos exemplos, no *Boosting* proposicional, é aqui mapeado em um processo iterativo de alteração da distribuição do grau dos vértices, gerando um classificador a cada iteração.

A seguir, considerando os classificadores produzidos a cada iteração e a Equação 2.2 é induzido o modelo final. O raciocínio básico é que um exemplo difícil de classificar, inicialmente, poderá ser classificado corretamente se mais relações desse exemplo com seus vizinhos forem consideradas, isto é, utiliza-se uma estratégia na qual, na dúvida, recorre-se a mais informação.



(a) (b) (c)

Figura 4.1: Aumento da distribuição de um vértice difícil de classificar

O processo de *Boosting* para grafos é resumido na figura 4.2. No primeiro passo, as bases de dados são divididas em um conjunto de exemplos para treino e outros para teste, conforme a Figura 4.2(a). A seguir, com cada conjunto de treino gera-se um grafo inicial (rede KNN inicialmente testado com

$k = 9$). Observa-se que na fase inicial da pesquisa ainda não tinha-se uma estratégia para definir o valor de k a ser adotado. Com esse grafo inicial e um algoritmo de classificação relacional, inicia-se o processo do *Boosting* gerando um conjunto novo de grafos com seus respectivos classificadores e erros associados, como apresentado na Figura 4.2(b). Tais erros definirão os pesos β dos respectivos classificadores. Finalmente, com os grafos gerados no passo anterior, pode-se classificar os vértices no conjunto de teste.

Observa-se que a estratégia de *boosting* utiliza iterativamente um classificador base para classificar exemplos de treino durante a fase de aprendizagem (observe a Figura 2.2). Na fase de classificação dos dados de teste também serão utilizados iterativamente os modelos gerados, agora com os seus respectivos pesos. Uma estratégia para a classificação é que para cada vértice a ser classificado procura-se por um vértice vs mais similar no conjunto de treino e utiliza as arestas e vizinhos de vs na sua classificação. Esta estratégia é utilizada devido ao fato que na etapa de treino é procurado o número ideal de vizinhos para a classificação correta dos exemplos de treino. Assim, parece razoável utilizar estes vizinhos para a classificação.

Na etapa de classificação dos dados de teste, combinam-se os grafos com os elementos do conjunto de teste, os classificadores relacionais e os pesos gerados para construir um classificador final. Este classificador final é usado para prever a classe de vértices no conjunto de teste, conforme a Figura 4.2(c).

Para entender com mais detalhe esta primeira tentativa de processo de *Boosting* em grafos é apresentado o Algoritmo 4.1 para a etapa de treino. O algoritmo foi baseado no algoritmo *Adaboost* proposto por Freund e Schapire (1995, 1996) adaptado para dados relacionais. Este algoritmo tem como entrada um grafo G_{tr} com n vértices de treino, um classificador relacional L e o número de iterações T. No começo, o algoritmo atribui pesos considerando o grau do vértice e o número total de arestas no grafo ($D_1(i) = \#arestas(V_{tr_i})/\#total_{arestas}$). Em seguida, o classificador L é treinado considerando o grafo G_{tr_t} e testado com os exemplos de treino. Os exemplos de treino mal classificados terão o seu peso D_t aumentado, enquanto que os exemplos bem classificados terão o seu peso diminuído. Além disso, é verificado o erro no conjunto de treino de cada classificador. Se o erro ultrapassar 0,5, então a técnica elimina o classificador correspondente e termina o algoritmo. O mesmo processo ocorre quando o erro é 0. O fato de utilizar um limiar superior de 0,5 é devido a que é bom criar um classificador *ensemble* usando classificadores individuais com taxa de erro menor que 0,5 (Dietterich, 2000a).

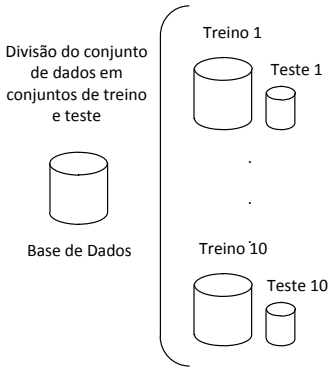
A seguir, considerando D_{t+1} é criado o novo grafo $G_{tr_{t+1}}$. Em relação a cada vértice deste grafo, o vértice incrementa seu número de arestas considerando seus vértices mais próximos ou diminui suas arestas de maneira aleatória. Finalmente, depois de executar as T iterações, o algoritmo terá como saída os modelos de classificação $(h_1, h_2, ..., h_T)$, os grafos $(G_{tr_1}, G_{tr_2}, ..., G_{tr_T})$, e os pesos $(\beta_1, \beta_2, ..., \beta_T)$, onde $\beta_t = (1 - e_t)/e_t$ e e_t é o erro do modelo na iteração t.

A etapa de classificação, apresentada no Algoritmo 4.2 que considera o modelo h_t e o grafo G_{tr_t}, é feita por votação com peso, sendo que cada classificador contribui com $\alpha_t = ln(\beta_t)$ votos. Este critério faz com que os classificadores com menores taxas de erro tenham pesos mais altos e vice-versa.

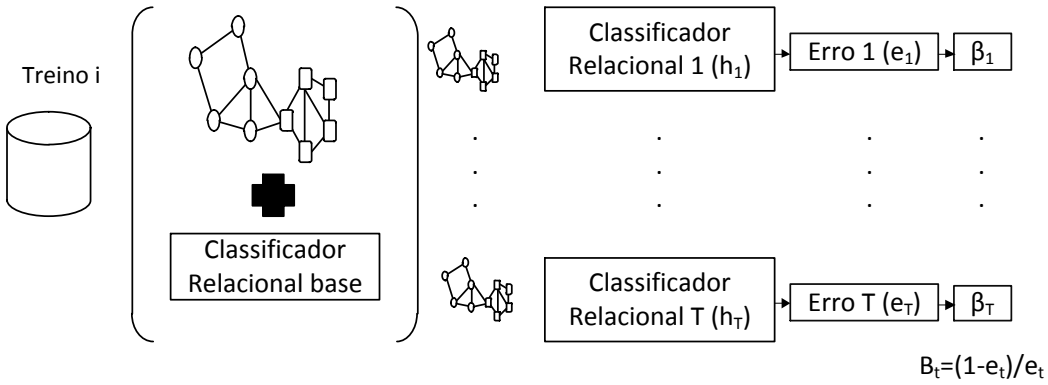
Finalmente, um exemplo detalhado desta proposta é apresentado na Figura 4.3 considerando 3 modelos ($T = 3$). Observe que os modelos são os novos grafos gerados com diferentes distribuições de conexões. Em primeiro lugar, na Figura 4.3(a) é apresentado o grafo inicial para a classificação. A seguir, na Figura 4.3(b) é induzido o primeiro modelo C_1 e seu erro associado é calculado e_1 resultando em 2 exemplos mal classificados (delimitados com um quadrado pontilhado no vértice). Em seguida, uma nova distribuição de arestas é gerada, usando essa nova distribuição são adicionadas/removidas arestas no grafo, o processo é mostrado na Figura 4.3(c). Em seguida, na Figura 4.3(d) um novo modelo C_2 é gerado com a nova distribuição e seu respectivo erro é calculado e_2. Finalmente, uma nova distribuição é gerada modificando o número de arestas do grafo e induzindo um novo modelo C_3 com um novo erro e_3, conforme as Figuras 4.3(e) e 4.3(f). Assim na etapa de classificação, o classificador resultante será gerado a partir dos modelos (grafos) induzidos, atribuindo maior ponderação aos com menor erro.

4.1.2 Experimentos Preliminares

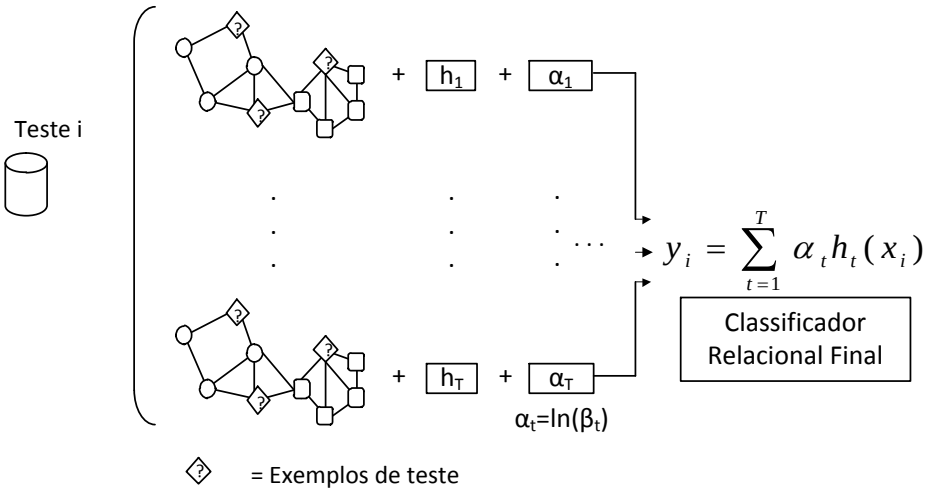
Como primeiro passo, ainda para entender o processo do *Boosting* tradicional foi implementado o algoritmo *AdaBoost*. Este algoritmo foi estudado usando uma extensão da ferramenta de visualização PEx (Paulovich et al., 2007) para visualizar iterativamente, o processo de alteração na distribuição de probabilidades dos exemplos. Para ilustrar este estudo foi utilizada a base de dados artificial bem conhecida, $Balance$ (Frank e Asuncion, 2010), a qual apresenta uma classe difícil e duas mais fáceis. A classe difícil (de cor verde) fica entre as outras duas misturando-se com essas. Este estudo é ilustrado na Figura 4.4 onde os pesos dos exemplos, a cada iteração, são representados pelo tamanho do exemplo. Um tamanho maior representa um peso maior e



(a)



(b)



(c)

Figura 4.2: Passos do processo de *Boosting* relacional. (a) Seleção de bases de dados de treino e teste, (b) Processo de *Boosting* para treino e (c) Processo de *Boosting* para classificação

Entrada:

- $G_{tr} = (V_{tr}, A_{tr})$: grafo dos vértices de treino.
- L: classificador baseado em grafos.
- T: número de iterações.

Saída:

- $h_1, h_2, ..., h_T$: T modelos gerados.
- $\beta_1, \beta_2, ..., \beta_T$: pesos dos T modelos gerados.
- $G_{tr_1}, G_{tr_2}, ..., G_{tr_T}$: T grafos.

Seja n o número de vértices de V_{tr}.
$D_1(i) \leftarrow$ (número de arestas do vértice i)/(número total de arestas dos vértices de treino do grafo), para cada um dos n exemplos.
$G_{tr_1} \leftarrow G_{tr}$.
para todo $t \leftarrow 1$ *até* T **faça**
  $h_t \leftarrow$ aplicar L aos exemplos usando o grafo G_{tr_t}.
  $e_t \leftarrow$ soma das distribuições $D_t(i)$ para exemplos mal classificados com h_t.
  se $e_t \geq 0.5$ *ou* $e_t = 0$ **então**
    Fim do algoritmo.
  fim
  $\beta_t \leftarrow (1 - e_t)/e_t$
  para todo $i \leftarrow 1$ *até* m **faça**
    se h_t *classifica mal o exemplo* i **então**
      $D_{t+1}(i) = D_t(i) * \beta_t$.
    senão
      $D_{t+1}(i) = D_t(i)$.
    fim
  fim
  Normalizar D_{t+1} tal que sua soma seja 1.
  $G_{tr_{t+1}} \leftarrow$ Criar um novo grafo usando o grafo G_{tr_t} considerando a distribuição D_{t+1}. Os exemplos que incrementarão sua distribuição, irão aumentar seu número de arestas (escolhendo os vértices mais próximos) e os que diminuirão, irão diminuir seu número de arestas de maneira aleatória (Um exemplo é apresentado na Figura 4.3).
fim
retorna $h_1, h_2, ..., h_T$, $G_{tr_1}, G_{tr_2}, ..., G_{tr_T}$ **e** $\beta_1, \beta_2, ..., \beta_T$

Algoritmo 4.1: Algoritmo de *Boosting* para classificadores relacionais baseados em grafos - Etapa de Aprendizagem.

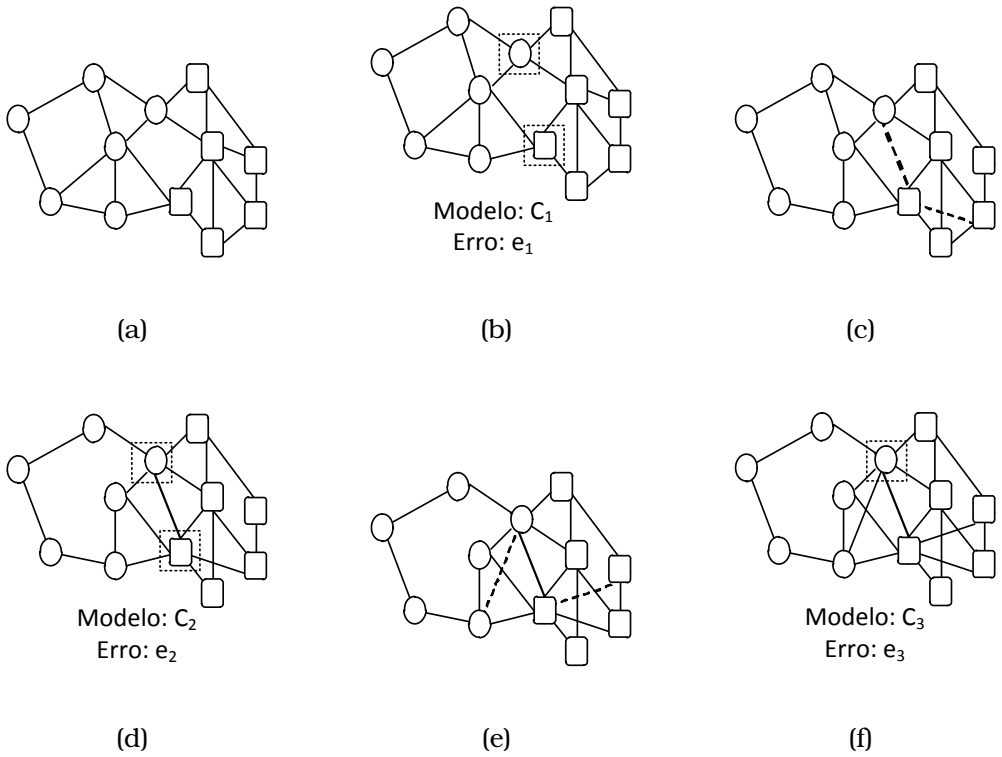


Figura 4.3: Exemplo da alteração de de distribuição das conexões nos modelos (grafos) no processo de *boosting* relacional proposto.

Entrada:

- $h_1, h_2, ..., h_T$: classificadores gerados no Algoritmo 4.1.
- $\beta_1, \beta_2, ..., \beta_T$: pesos gerados no Algoritmo 4.1.
- $G_{tr_1}, G_{tr_2}, ..., G_{tr_T}$: grafos gerados no Algoritmo 4.1.
- T: número de modelos gerados.
- e: exemplo a classificar.

Saída:

- c: classe estimada do exemplo e.

atribuir o peso de 0 para todas as classes.
para todo $t \leftarrow 1$ *até* T **faça**
  $\alpha_t = ln(\beta_t)$.
  Adicionar α_t ao peso da classe predita pelo modelo: $h_t(G_{tr_t}, e)$.
fim
$c \leftarrow$ determinar classe com maior peso.
retorna c

Algoritmo 4.2: Algoritmo de *Boosting* classificadores relacionais baseados em grafos - Etapa de Classificação.

um tamanho pequeno um peso menor. A cada iteração, pode-se ver que o algoritmo se concentra nos exemplos difíceis de classificar (os exemplos da classe verde).

A seguir, foram realizados alguns experimentos com dados atributo-valor extraídos do repositório UCI (Frank e Asuncion, 2010), e transformados para o formato de grafo não dirigido usando uma rede KNN. Outra estratégia baseada em uma rede de similaridade pode criar um grafo dirigido. O algoritmo de classificação de grafos empregado foi o classificador relacional baseado nos vizinhos com votação ponderada ($wvrn$) (Macskassy e Provost, 2004), o qual foi empregado no processo de *Boosting* para classificação baseada em grafos. Os resultados são apresentados na Tabela 4.1.

Para a análise dos resultados, foi utilizado o teste de Wilcoxon (Demšar, 2006),o qual determinou não existir diferença estatística entre classificadores bases e suas versões *boosting*. Posteriormente, foi analisada esta primeira versão e observou-se que uma das razões possiveis deveu-se ao fato do uso de grafos não dirigidos. Já que quando se adiciona uma aresta, não só está adicionando esta aresta ao vértice analisado mas também a outro mais. O mesmo processo acontece quando se elimina arestas, não respei-

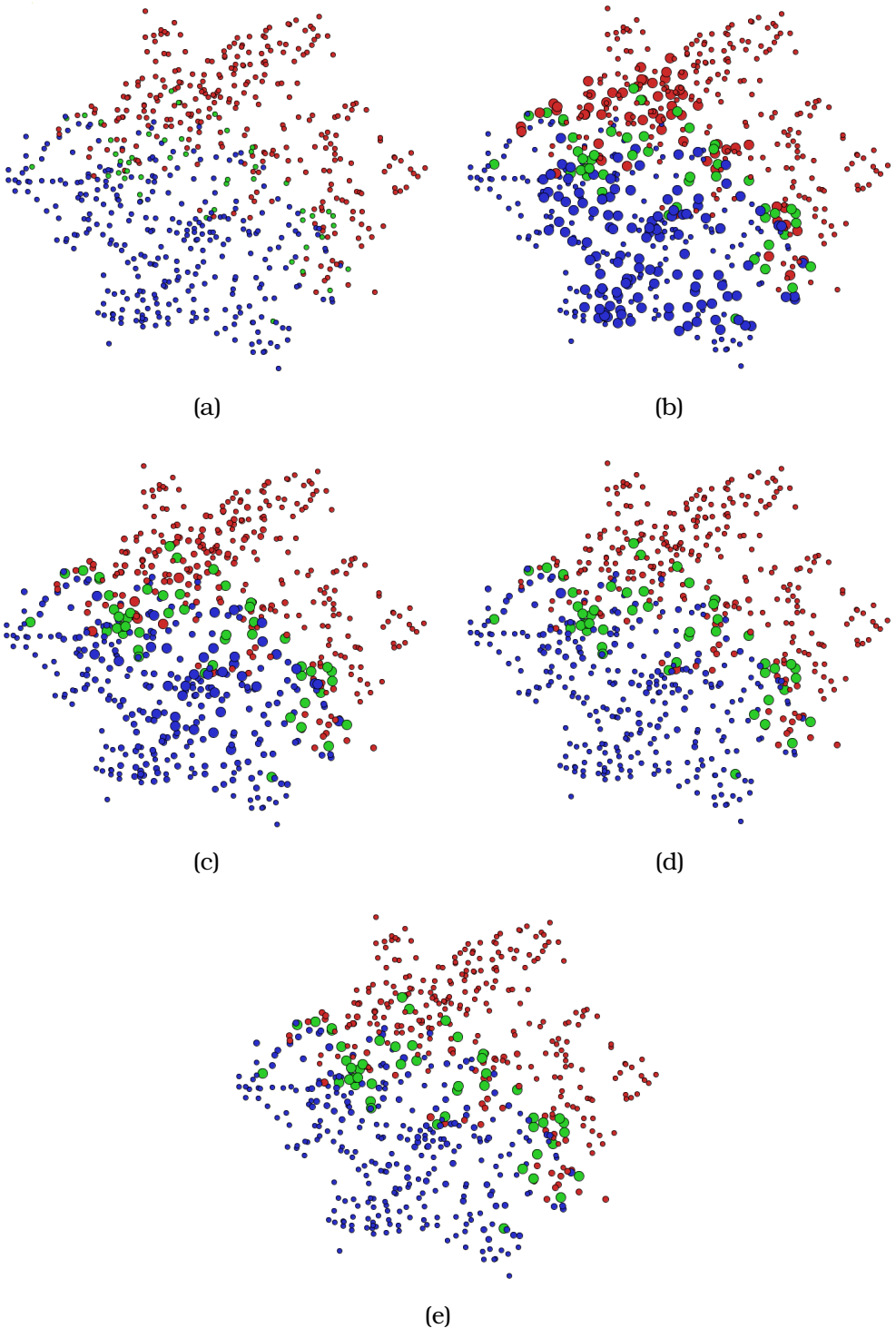


Figura 4.4: Cinco iterações do algoritmo de *Adaboost* em dados proposicionais, mostrando a alteração na distribuição nos exemplos. Os exemplos mais difíceis estão proporcionalmente aumentados.

	wvrn		*Boosting* usando wvrn	
Base de dados	média	desvio padrão	média	desvio padrão
Glass	0.63	0.14	0.63	0.14
Statlog	0.58	0.14	0.58	0.14
Wine	0.74	0.11	0.74	0.11
Iris	0.97	0.05	0.97	0.05
Blood	0.57	0.13	0.57	0.13
Lung	0.67	0.36	0.65	0.36
Haberman	0.74	0.06	0.73	0.07

Tabela 4.1: Média e desvio padrão da acurácia para o classificador wvrn e o classificador *Boosting* para grafos não dirigidos usando wvrn em diferentes bases de dados.

tando portanto a distribuição dos exemplos. Assim, pode-se estar melhorando a classificação dos vértices errados, porém piorando a de outros.

Foi desenvolvida uma ferramenta para entender a mudança do grafo inicial ao longo do processo do *Boosting* com as seguintes funções: ver os exemplos classificados erroneamente em cada iteração, ver a classe determinada por cada classificador gerado, gerar arquivos para sua análise gráfica no PEx (Paulovich et al., 2007), gerar tabelas com a taxa de acerto, entre outros. Adicionalmente, foram adicionadas algumas novas características a ferramenta PEx para visualização de grafos.

Na Tabela 4.2, é apresentado o processo de *Boosting* em grafos não dirigidos com o número de exemplos classificados erroneamente e o peso atribuído a cada classificador gerado. Nesta tabela, pelo uso de um grafo não dirigido o número de exemplos mal classificados em cada iteração aumenta. Então, o *boosting* relacional não melhora o classificador base, devido ao fato que este algoritmo alcança rapidamente um erro $e_t >= 0,5$, gerando apenas um ou dois classificadores no *ensemble*. No entanto, na Tabela 4.3 é apresentado o mesmo experimento com uma rede de similaridade (usando um grafo dirigido) e mostra que o número de exemplos classificados erroneamente diminui a partir da primeira iteração e permite gerar um maior número de classificadores individuais no *ensemble*.

A seguir, foi feito um estudo do processo de *Boosting* para grafos utilizando a ferramenta PEx adaptada, com a base de dados *Iris* (Frank e Asuncion, 2010). Este estudo é apresentado nas figuras 4.5 e 4.6, no qual se percebe um comportamento similar ao do *Boosting* tradicinal. Os exemplos mais difíceis de classificar aumentam seu peso, no caso do grafo seu número de conexões. Assim, em cada iteração o algoritmo concentra-se nos exemplos mais difíceis

	Iteração 1		Iteração 2		Iteração 3	
Base de dados	N° exemplos errados	Erro/peso	N° exemplos errados	Erro/peso	N° exemplos errados	Erro/peso
Glass	51	0.26/2.85	67	0.53/-	-	-
Statlog	41	0.51/-	-	-	-	-
Wine	44	0.26/2.85	65	0.58/-	-	-
Iris	7	0.05/19	26	0.51/-	-	-
Blood	62	0.34/1.94	62	0.48/1.08	66	0.51/-
Lung	7	0.36/1.78	6	0.46/1.17	8	0.53/-
Haberman	78	0.24/3.17	90	0.48/1.08	103	0.52/-

Tabela 4.2: Tabela do processo de *Boosting* em grafos não dirigidos

	Iteração 1		Iteração 2		Iteração 3	
Base de dados	N° exemplos errados	Erro/peso	N° exemplos errados	Erro/peso	N° exemplos errados	Erro/peso
Glass	49	0.25/3	43	0.41/1.44	43	0.49/1.04
Statlog	37	0.44/1.27	37	0.51/-	-	-
Wine	40	0.25/3	33	0.39/1.59	36	0.49/1.04
Iris	7	0.05/19	4	0.28/2.57	4	0.49/1.04
Blood	62	0.38/1.63	60	0.48/1.08	62	0.51/-
Lung	11	0.45/1.22	11	0.51/-	-	-
Haberman	79	0.28/2.57	62	0.37/1.70	73	0.54/-

Tabela 4.3: Tabela do processo de *Boosting* em grafos dirigidos

de classificar. No entanto, a classificação não melhorou com o uso do grafo dirigido, conforme observa-se na Tabela 4.4.

	wvrn		*Boosting* usando wvrn	
Base de dados	média	desvio padrão	média	desvio padrão
Glass	0.65	0.11	0.65	0.11
Statlog	0.62	0.13	0.62	0.13
Wine	0.75	0.11	0.75	0.11
Iris	0.96	0.06	0.96	0.06
Blood	0.44	0.16	0.45	0.16
Lung	0.52	0.23	0.52	0.23
Haberman	0.68	0.12	0.68	0.11

Tabela 4.4: Média e desvio padrão da acurácia para o classificador wvrn e o classificador *Boosting* para grafos dirigidos usando wvrn em diferentes bases de dados

Uma possível razão para os resultados desta segunda versão pode ser, devido ao fato que tanto no *Boosting* usando grafos não dirigidos (rede KNN) quanto dirigidos (rede de similaridade) o primeiro classificador gerado sempre tem um maior peso que a soma dos outros classificadores, atribuindo assim como resultado final em geral a classe definida pelo primeiro classificador.

Continuando a investigação para procurar o porquê do mapeamento da estratégia de *boosting* para classificadores baseados em grafos não estar dando certo, observou-se que, considerando o trabalho de García-Pedrajas e Ortiz-Boyer (2009) e o grafo representado por uma rede KNN ou rede de similaridade, métodos *Ensemble*, particularmente o *boosting*, consideram a instabilidade de classificadores para melhorar seu desempenho. No entanto, o algoritmo KNN é bastante estável em relação à amostragem. Assim, esses métodos propostos falham em sua tentativa de melhorar o desempenho do classificador base, pois comportam-se de forma similar ao classificador KNN.

Assim, decidiu-se usar um enfoque iterativo e selecionar apenas o classificador final no algoritmo de treino. A seguir, o conjunto de treino foi dividido em dois conjuntos. O primeiro para a etapa de treino propriamente dita e o segundo, denominado conjunto de validação, para determinar o melhor desempenho do algoritmo em relação ao valor do número de vizinhos (k). Assim, foi determinado o valor máximo de $k = 51$ como um limiar máximo aplicável na grande maioria dos domínios testados. Isto é, em nenhum caso obteve-se uma rede KNN melhor com k superior a esse valor. Estas decisões levaram ao classificador KNN Adaptativo Baseado em Grafos (A-KNN), que conseguiu nos casos avaliados, melhorar significativamente ou pelo menos manter a quali-

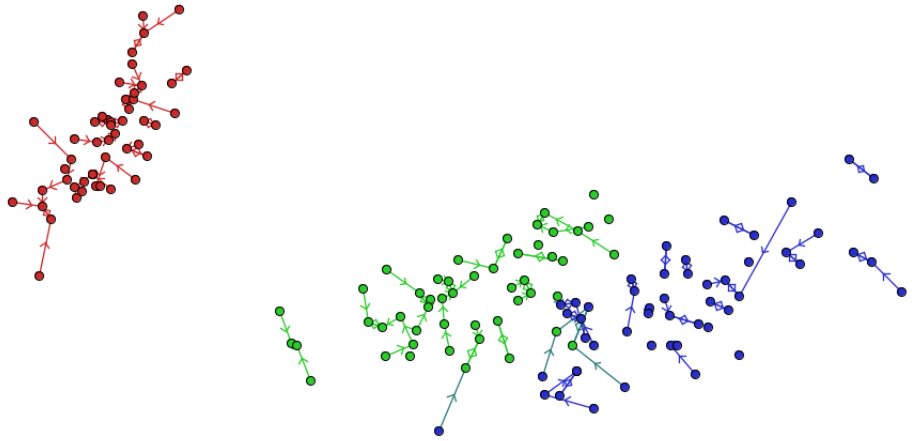
dade dos classificadores bases.

4.2 KNN Adaptativo Baseado em Grafos (A-KNN)

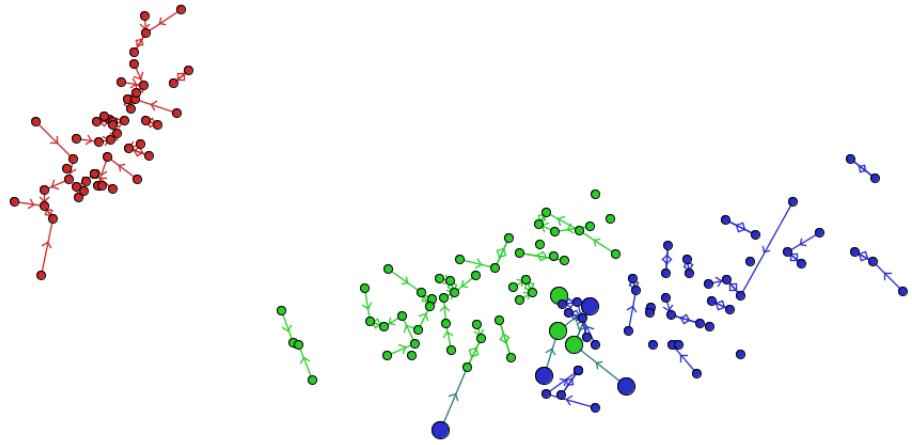
O algoritmo proposto KNN Adaptativo Baseado em Grafos poder ser classificado como um *ensemble* dependente (veja a seção 2.2.2), especificamente como um *incremental batch learning*. Este classificador utiliza uma representação de uma rede de similaridade e um algoritmo de *Boosting* baseado em grafos. Seu objetivo principal é melhorar o desempenho do classificador relacional baseados nos vizinhos com votação ponderada (*wvrn*) (Macskassy e Provost, 2004) mapeando o *Boosting* proposicional (Freund e Schapire, 1996) em um relacional.

A lógica do algoritmo A-KNN é alterar a distribuição de conexões em uma rede de similaridade em vez de alterar a distribuição dos exemplos como é feito no *Boosting* proposicional. Assim, o modelo final corresponde, de certa forma, a um KNN capaz de usar diferentes valores de k (entre 1 e um $k_{max} = 51$) para diferentes regiões no espaço dos dados. Por exemplo, nas Figuras 4.7(a), 4.7(b) e 4.7(c) são ilustradas as mudanças na distribuição de conexões durante as 3 primeiras iterações em uma rede de similaridade gerada a partir do conjunto de dados, já comentado na seção anterior, *Balance* (Frank e Asuncion, 2010). O tamanho do círculo representa o grau (número de conexões) dos vértices. Na Figura 4.7(d), não são mostradas as arestas a fim de manter a figura limpa e mostrar como os exemplos difíceis (círculos verdes grandes) são claramente os mais difíceis, pois estão misturados entre as duas outras classes.

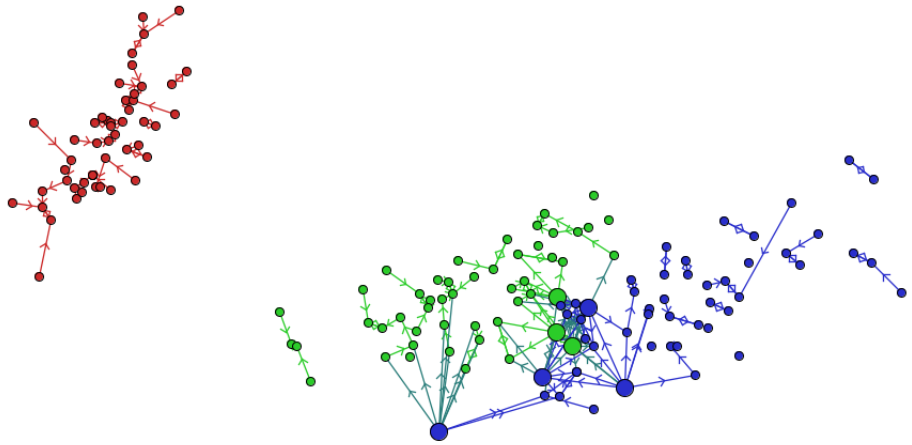
O A-KNN consiste em duas fases: a fase de treino e a de classificação. Na fase de treino, como apresentada na Figura 4.8 e no Algoritmo 4.3, primeiramente, cria-se uma rede de similaridade a partir dos dados de treino $G_{tr}(V, A)$, definindo $k_{max} = 51$. Efetivamente, esta rede inicial fornece o conjunto de arestas A possível a partir do qual serão escolhidas as arestas a serem utilizadas ($A_{final} \in A$) na rede final. Ou seja, a rede KNN com $k = k_{max}$, simplesmente serve para "disponibilizar" o conjunto de arestas que poderão ser usadas na geração dos modelos no processo de *boosting*. Depois disso, usando V e A, iterativamente constrói-se redes de similaridade (k variando de 1 até k_{max}, com incrementos de dois). Em cada iteração, todos os exemplos são classificados usando suas conexões e o classificador relacional baseado nos vizinhos com votação ponderada (*wvrn*) (usando o mesmo peso para todas as arestas). Exemplos mal classificados poderão selecionar mais arestas de A. Estas ares-



(a)

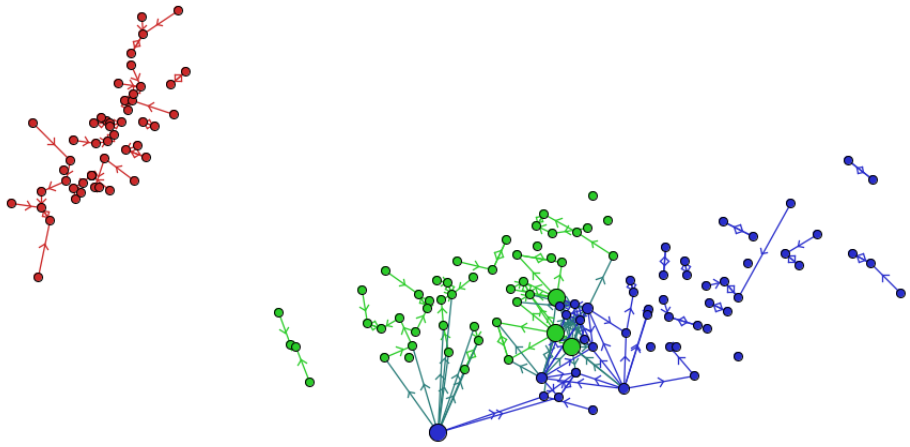


(b)

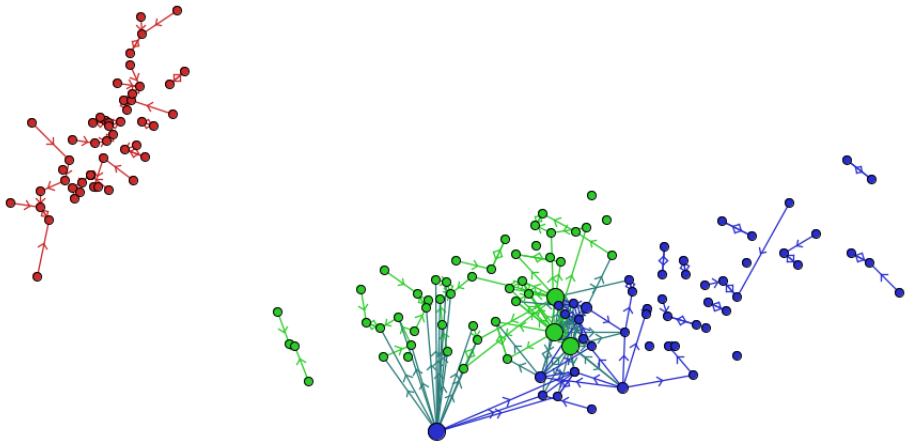


(c)

Figura 4.5: Iterações 1, 2 e 3 da distribuição das arestas na adaptação do algoritmo de *Boosting* para grafos. As arestas estão representadas por uma seta dirigida, a qual conecta um vértice v_i (inicio) com seus vizinhos v_j (fim).



(d)



(e)

Figura 4.6: Iterações 4 e 5 da distribuição das arestas na adaptação do algoritmo de *Boosting* para grafos. As arestas estão representadas por uma seta dirigida, a qual conecta um vértice v_i (inicio) com seus vizinhos v_j (fim).

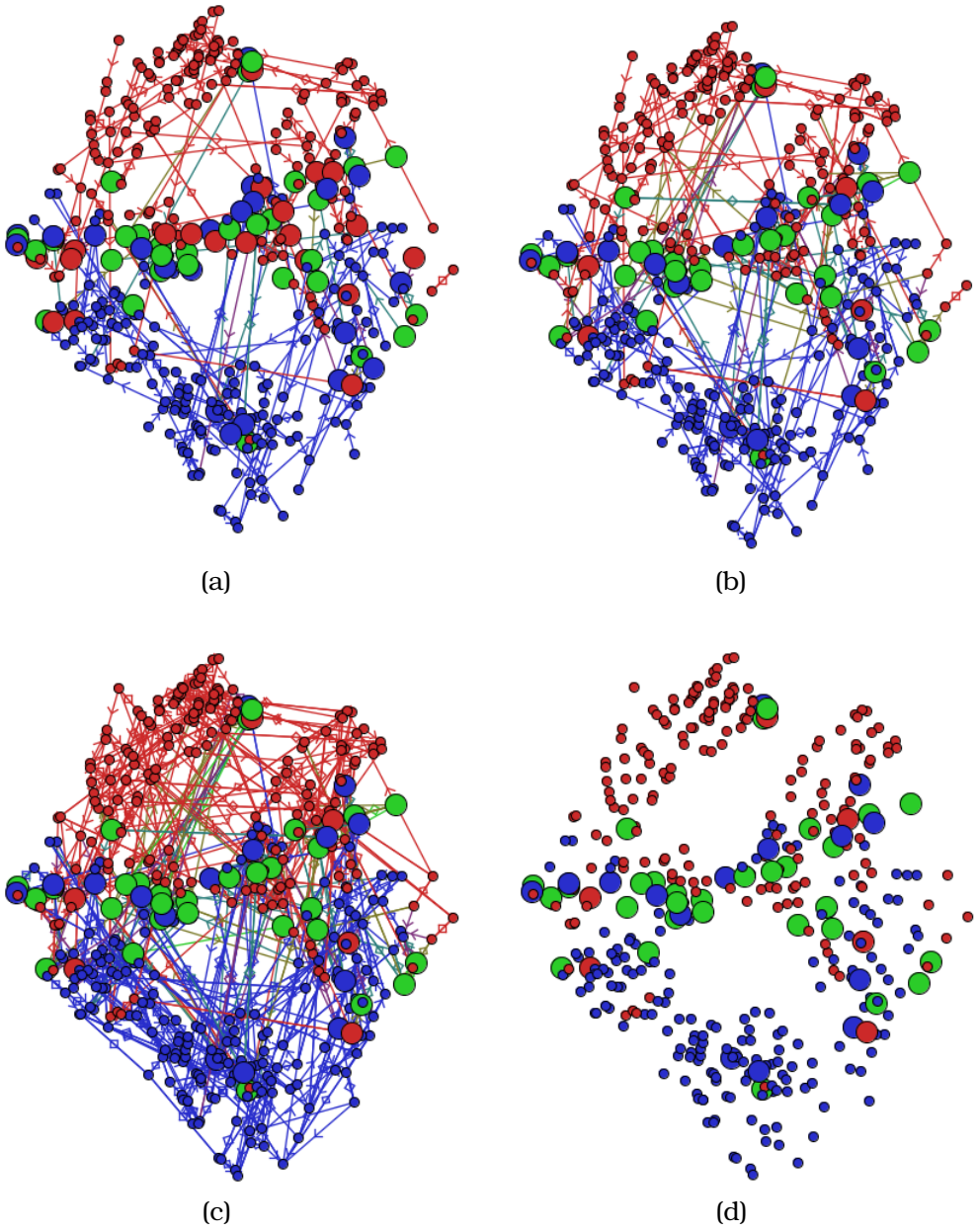


Figura 4.7: Progresso da distribuição de conexões no conjunto de dados *Balance*. (a) Iteração 1. (b) Iteração 2. (c) Iteração 3. (d) Iteração Final.

tas adicionais são escolhidas obedecendo o *ranking* de similaridade com os 51 vizinhos. São usadas primeiro as arestas que conectem aos vizinhos mais próximos.

Durante esse processo, a situação ideal é que exemplos classificados erroneamente ou não classificados (estado=0) terão seu número de ligações aumentado. Em alguma etapa deste aumento, o exemplo pode ser classificado corretamente (estado=1). Continuando aumentando o número de ligações o exemplo pode voltar a ficar errado (estado=0). A idéia do algoritmo é procurar um valor estável de k (na faixa de classificação correta) para cada exemplo no treino, este valor poderá ser utilizado na classificação de exemplos de teste. Esta situação é ilustrada na iteração 2 da Figura 4.8 em relação ao vértice com uma conexão.

A seguir, esse processo termina quando o número de iterações T é alcançado ($T = k_{max}/2$) ou o grafo gerado G_{i+1} em cada iteração não apresenta mudanças. Finalmente, considerando os exemplos de treino E_{tr} e a evolução dos seus $estados$ é gerado o modelo h_f como resultado da etapa de aprendizagem.

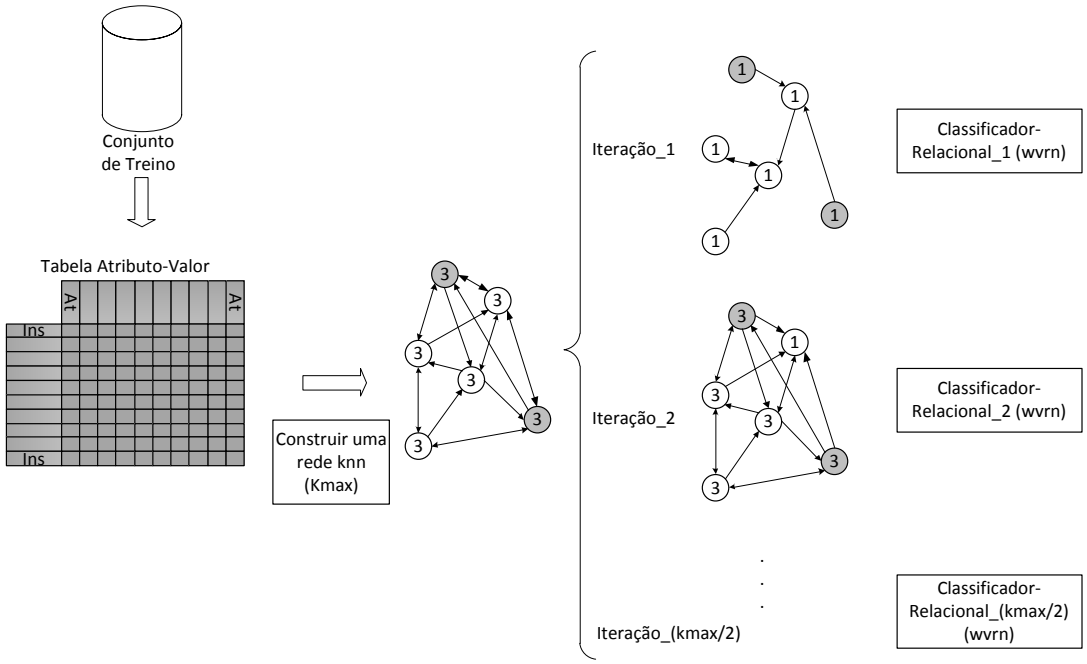


Figura 4.8: Etapa de treino do algoritmo A-KNN. As cores representam as classes dos exemplos. Além disso, o número em cada vértice representa seu número de conexões.

Observe que, variando o valor de k de 0 até k_{max} (como ilustrado na Figura

Entrada:

- E_{tr}: conjunto de dados de treino.
- L: classificador base.
- k_{max}: Valor dos vizinhos por instância.

Saída:

- h_f: Classificador gerado no processo de treino.

Algoritmo:
$pG \leftarrow$ Criar uma rede de similaridade usando $k = k_{max}$ com o conjunto E_{tr}.
$G_1 \leftarrow$ Criar uma rede de similaridade usando $k = 1$ com o conjunto E_{tr}.
para $i \leftarrow 1$ *até* $tamanho(E_{tr})$ **faça**
  $d_i \leftarrow 1$ `// grau do vertice` i.
  $estados_i \leftarrow 0$.
fim
$T \leftarrow k_{max}/2$ `// Número de iterações.`
para $i \leftarrow 1$ *até* T **faça**
  $h \leftarrow$ treinarClassificador(L,E_{tr},G_i).
  $(d, estados) \leftarrow$ atualizarConexões(h,E_{tr},d,$estados$). `// como explicado na seção 4.2.`
  $G_{i+1} \leftarrow$ criarGrafo(E_{tr},d,pG).
  se $G_{i+1} == G_i$ **então**
    Termina.
  fim
fim
$h_f \leftarrow$ CriarClassificadorFinal(E_{tr},$estados$) `// considerando a Figura 4.9.`
retorna h_f

Algoritmo 4.3: A-KNN - Algoritmo de Aprendizagem.

4.9), três casos podem ocorrer: (a) a classificação de um vértice pode ser alterada do estado 0 a 1 (quando $k = k_a$), e finalmente voltar ao estado 0 (quando $k = k_c$), (b) a classificação é mantida no estado 1, após $k = k_b$, ou (c) mantida no estado 0. Assim, nos casos (a) e (b) pode-se escolher qualquer valor de k que conduza ao estado 1 para cada vértice da rede final, e para (c) não importa o valor de k. Isto é, os valores adequados de k para os vértices em $G(V, A_{final})$, nos experimentos descritos na próxima seção, são calculados por $(k_a + k_c)/2$, $(k_b + k_{max})/2$, k_{max} para cada caso, respectivamente. Assim, note que a situação ideal para a classificação, seria que todos os exemplos apresentaram um comportamento como o caso (a).

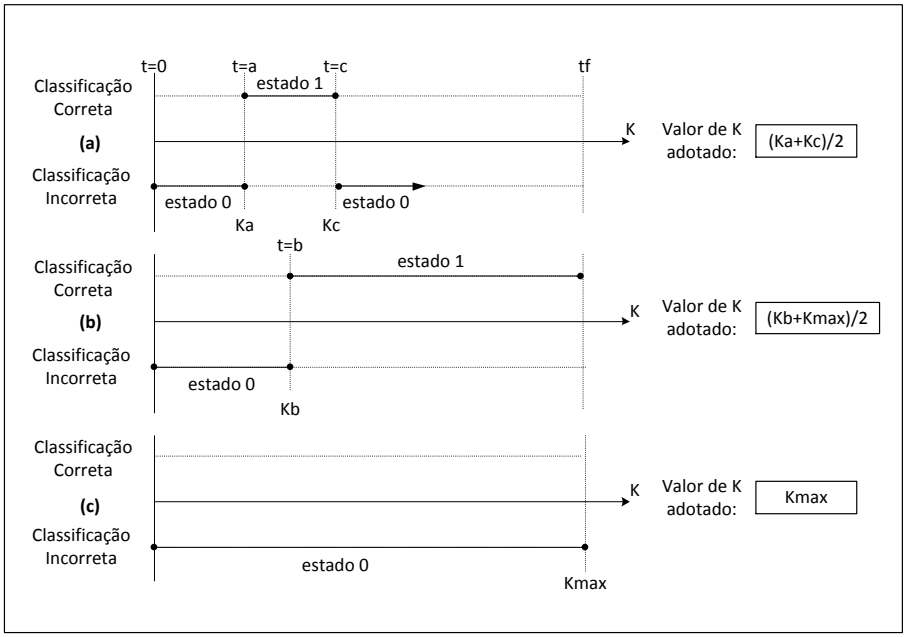


Figura 4.9: Valores possíveis de k adotados no classificador final dado um vértice.

Na fase de classificação, conforme resumido no Algoritmo 4.4, considerando a rede final. Em primeiro lugar, procura-se o vizinho mais próximo (vp) da instância de teste e, e em segundo lugar, considerando o número de arestas do vizinho mais próximo, k_{vp}, classifica-se a instância de teste olhando para seus k_{vp} vizinhos mais próximos. Observa-se que para exemplos em regiões difíceis um maior número de vizinhos será utilizado e para regiões fáceis um número pequeno.

Por exemplo, na Figura 4.10, o exemplo de teste é representado pelo símbolo "?". Em primeiro lugar, o exemplo "?" procura seu vizinho mais próximo vp. A

seguir, é determinado o número de arestas de vp ($k_{vp} = 3$), conforme a Figura 4.10 (a). Assim, procura-se os 3 vizinhos mais próximos do exemplo "?", conforme a Figura 4.10 (b). Estes exemplos vizinhos estão conectados ao exemplo "?" com linhas pontilhadas. Finalmente, considerando as classes destes vizinhos é determinada a classe do exemplo "?", conforme a Figura 4.10 (c).

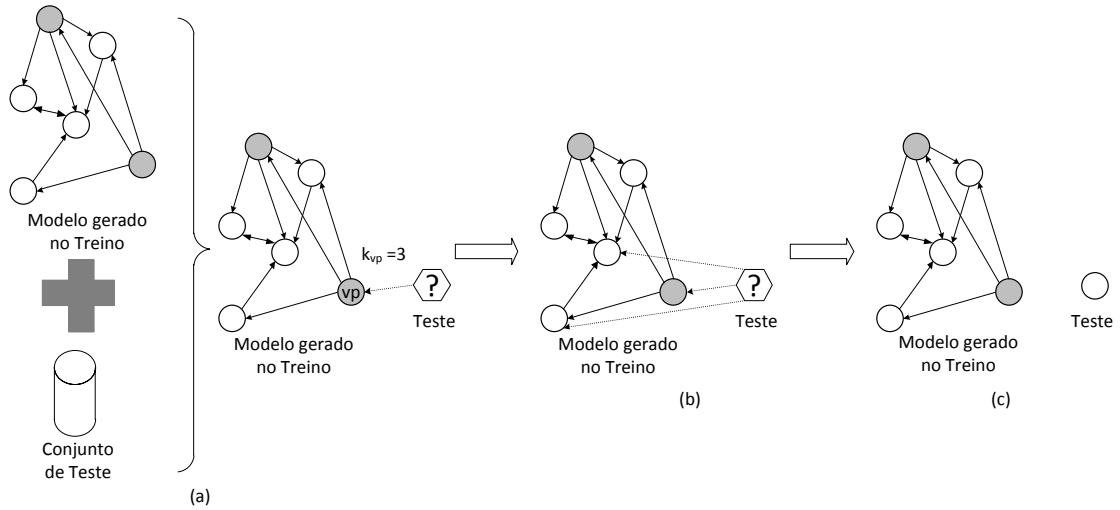


Figura 4.10: Etapa de Classificação. As cores representam as classes nos exemplos e o símbolo "?" representa um exemplo de teste.

Entrada:

- e: exemplo para classificar.
- h_f: classificador gerado no processo de treino.

Saída:

- c: classe predita.

Algoritmo:
$vp \leftarrow$ Encontre o vizinho mais próximo do exemplo e.
$k_{vp} \leftarrow$ Obter o número de conexões de vp considerando o grafo de h_f.
$c \leftarrow$ Classificar e usando seus k_{vp} vizinhos mais próximos.
retorna c

Algoritmo 4.4: A-KNN - Algoritmo de Classificação.

4.3 *Ensembles* baseados em grafos

Nesta subseção, são apresentados os classificadores *ensembles* desenvolvidos no contexto de dados no formato de grafo. Estes *ensembles* melhoram a acurácia de classificadores baseados em grafos ou nos casos que não melhoraram mantiveram-na.

Estes classificadores consideram as seguintes etapas: (i) amostragem do conjunto de vértices; (ii) criação de um novo grafo; (iii) indução do modelo de classificação (h_t) considerando o novo grafo e o classificador L; e finalmente, (iv) combinação dos modelos gerados ($h_1, h_2, ..., h_T$) para classificar os exemplos de teste. Este processo é apresentado na Figura 4.11.

Nesse contexto, foram considerados os seguintes esquemas propostos: (1) *Bagging* baseado em grafos e (2) *Cross-Validated Committees* baseados em grafos. A seguir, é apresentado uma descrição desses esquemas.

4.3.1 *Bagging* baseado em grafos (BG)

O BG apresenta 2 etapas: a fase de treino e de classificação. Na fase de treino, considerando o Algoritmo 4.5, amostra-se com reposição n (tamanho do conjunto de treino) vértices do conjunto de treino e armazena-se em V_{tr_t}. Em seguida, cria-se o grafo G_{tr_t} considerando o grafo de treino (G_{tr}) e V_{tr_t}. A seguir, é induzido o modelo h_t considerando o classificador L e o grafo G_{tr_t}. O modelo h_t é armazenado. Finalmente, o algoritmo retorna um conjunto de modelos $h_1, h_2, ..., h_T$.

Na fase de classificação, o modelo final prediz a classe de um exemplo de teste (x_i) considerando a Equação 4.1. Nesta equação, M_j denota o classificador j e $P_{M_j}(y_i = c_k|N_i)$ denota a probabilidade de y_i ser da classe c_k dada sua vizinhança N_i.

$$P(y_i = c_k|N_i) = \underset{c_k \in dom(classes)}{\operatorname{argmax}} \left(\sum_j P_{M_j}(y_i = c_k|N_i) \right) \tag{4.1}$$

Neste esquema, foram desenvolvidos 3 algoritmos mudando a etapa de criação do grafo. Estes algoritmos são (1) *Bagging* baseado em Grafos com Duplicação de Arestas, (2) *Bagging* baseado em Grafos com Pesos e (3) *Bagging* baseado em Grafos com Remoção de Vértices Duplicados.

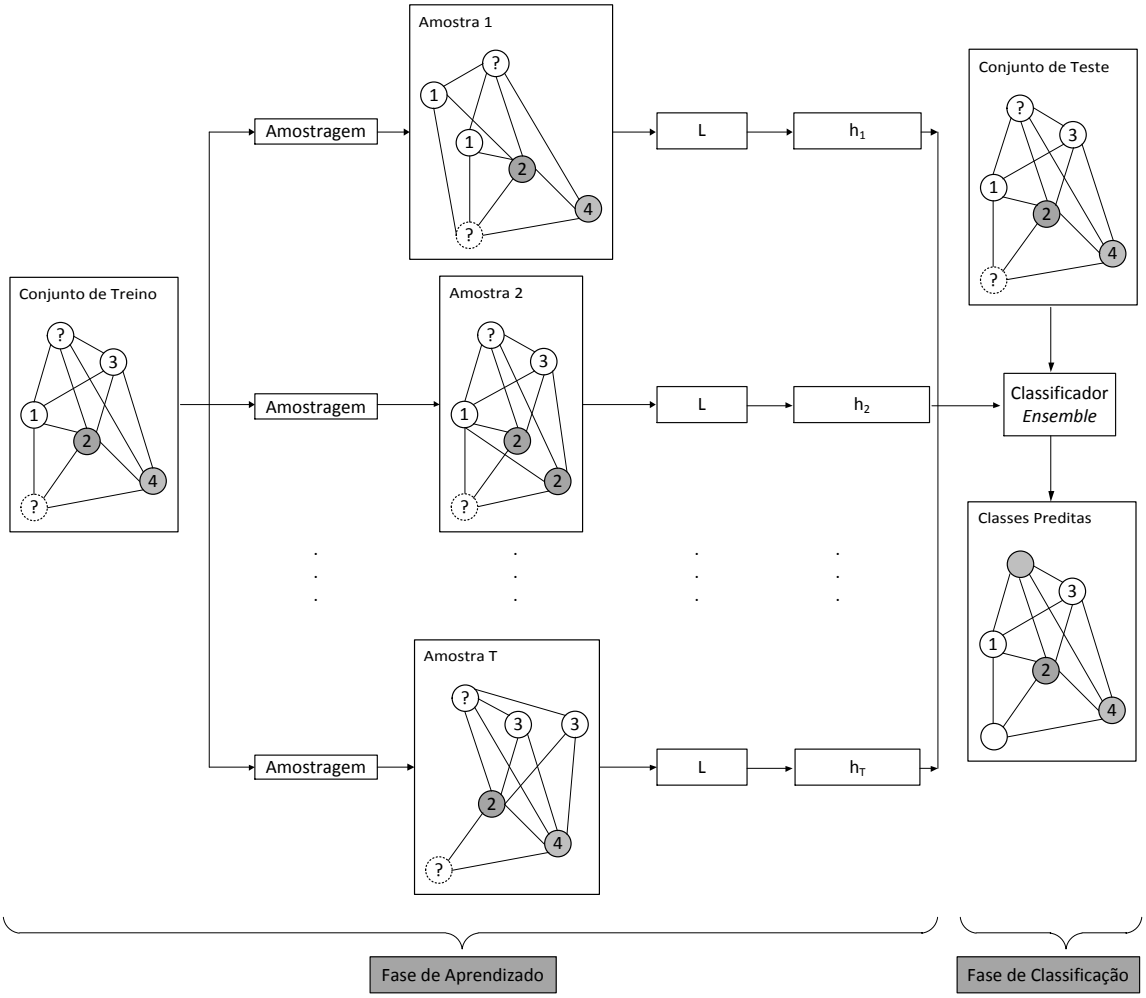


Figura 4.11: Processo geral para a criação de *ensembles* em dados no formato de grafo. As cores representam as classes dos exemplos e o símbolo "?" representa um exemplo de teste. Além disso, T representa o número de amostras; L o algoritmo de aprendizado e h_i os modelos gerados usando L.

Entrada:

- $G_{tr} = (V_{tr}, A_{tr})$: Conjunto de Treino no formato grafo.
- L: Classificador relacional base.
- T: Número de iterações.

Saída:

- $h_1, h_2, ..., h_T$: T modelos gerados.

Algoritmo:
Seja n o número de vértices de V_{tr}.
para $t \leftarrow 1$ *até* T **faça**
    $V_{tr_t} \leftarrow$ Amostrar n vértices com reposição dos vértices V_{tr}.
    $G_{tr_t} \leftarrow$ criarGrafo(G_{tr},V_{tr_t}).
    $h_t \leftarrow$ Aplicar o algoritmo de aprendizado (L) ao grafo (G_{tr_t}).
    Armazenar o modelo h_t.
fim
retorna $h_1, h_2, ..., h_T$

Algoritmo 4.5: *Bagging* baseado em grafos - Etapa de Aprendizagem.

Bagging baseado em Grafos com Duplicação de Arestas (BGDA)

Na etapa de criação do grafo no algoritmo 4.5, os vértices que aparecem mais de uma vez repetirão suas conexões com seus vizinhos e vértices que não aparecem terão suas conexões removidas e eliminadas. Por exemplo, na Figura 4.12, considerando a amostra 1, o vértice 1 aparece 2 vezes e o vértice 2 foi removido.

Bagging baseado em Grafos com Pesos (BGP)

Considerando o algoritmo 4.5, na etapa de criação do grafo, os vértices que aparecem mais de uma vez duplicarão os pesos das suas arestas e os vértices que não aparecem terão suas arestas removidas e eliminadas. Na Figura 4.13, assume-se que o grafo original contém arestas com peso 1. Por exemplo, na amostra 1 as arestas do vértice 1 dobraram seu peso e o vértice 2 foi removido.

Bagging baseado em Grafos com Remoção de Vértices Duplicados (BGRVD)

Considerando o algoritmo 4.5 e a Figura 4.14, amostra-se com reposição n vértices de V_{tr} e armazena-se em V_{tr_t}. A seguir, considerando V_{tr_t}, os vértices repetidos são removidos gerando o conjunto de vértices C_{tr_t}. Em seguida, considerando o conjunto de vértices C_{tr_t} seleciona-se um subgrafo G_{tr_t} do grafo

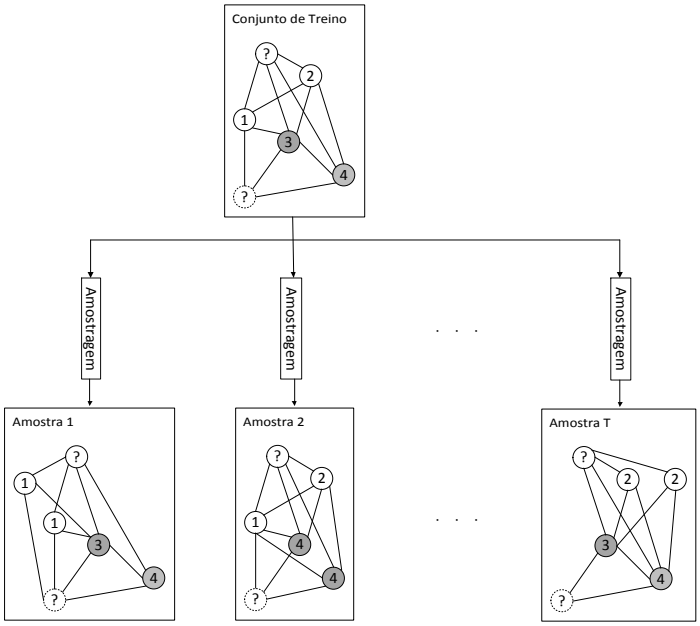


Figura 4.12: Amostragem de vértices no algoritmo BGDA.

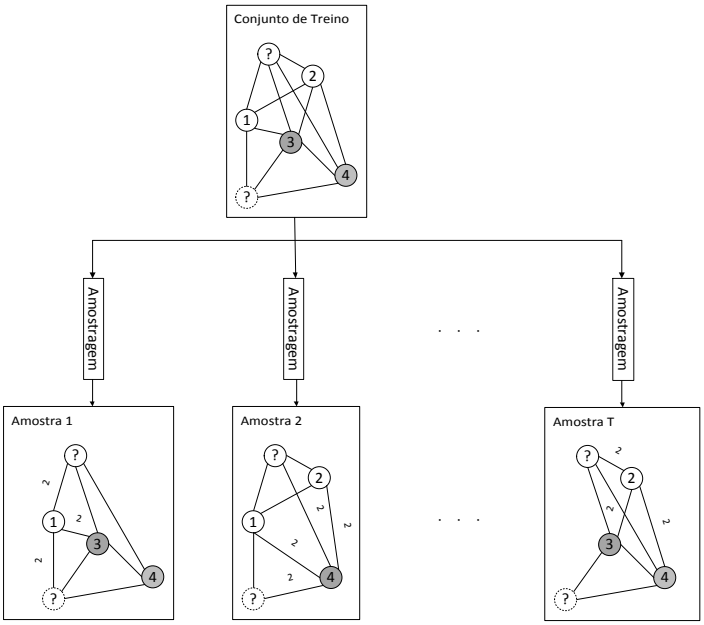


Figura 4.13: Amostragem de vértices no algoritmo BGP.

de treino G_{tr}, conforme a Figura 4.14. Por exemplo, na amostra 2, o subgrafo com os vértices $C_{tr_2} = \{1, 2, 4\}$ foi selecionado.

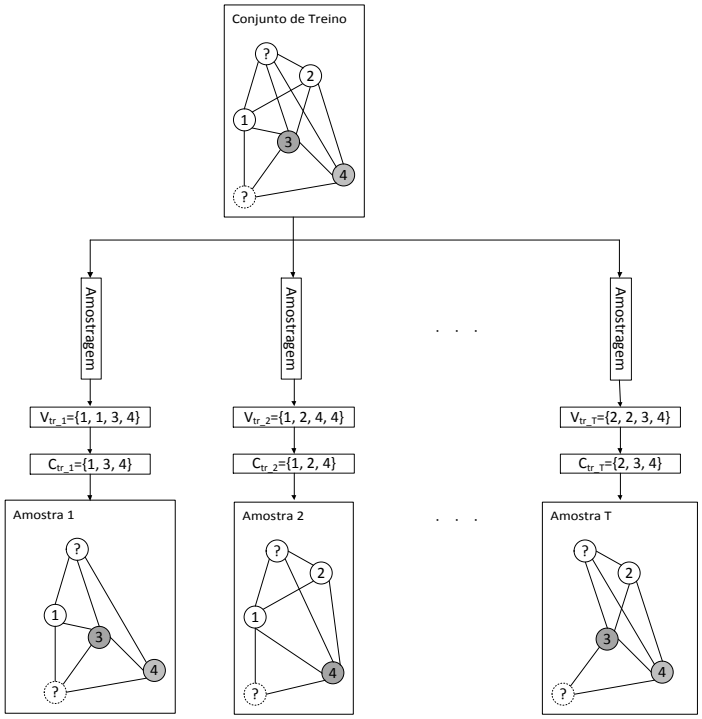


Figura 4.14: Amostragem de vértices no algoritmo BGRVD.

4.3.2 *Cross-Validated Committees* baseados em Grafos (CVCG)

O algoritmo CVCG também apresenta 2 etapas: a fase de treino e de classificação. Na fase de treino, resumida no Algoritmo 4.6, em primeiro lugar, o conjunto de vértices de treino V_{tr} é dividido em T conjuntos ($cv_1, cv_2, ..., cv_T$), isto é, particionar o conjunto de vértices V_{tr} em T conjuntos disjuntos de tamanho igual. Essa partição é feita de maneira estratificada, considerando as classes dos vértices. Em segundo lugar, já em cada iteração do algoritmo, é desconsiderado um desses conjuntos cv_t para gerar um novo conjunto de treino e armazená-lo em V_{tr_t}. A seguir, na criação do grafo apresentado na Figura 4.15, considerando os vértices V_{tr_t} é selecionado um sub-grafo de G_{tr} e armazenado em G_{tr_t}. Em seguida, é induzido o modelo h_t considerando o classificador L e o grafo G_{tr_t}. Finalmente, o algoritmo armazena o modelo h_t e retorna um conjunto de modelos $h_1, h_2, ..., h_T$.

Na fase de classificação, é usado o mesmo raciocínio do esquema de BG

considerando a Equação 4.1.

Entrada:

- $G_{tr} = (V_{tr}, A_{tr})$: Conjunto de Treino no formato grafo.
- L: Classificador relacional base.
- T: Número de iterações.

Saída:

- $h_1, h_2, ..., h_T$: T modelos gerados.

Algoritmo:
$(cv_1, cv_2, ..., cv_T) \leftarrow$ Dividir os vértices V_{tr} em T conjuntos disjuntos de maneira estratificada.
para $t \leftarrow 1$ *até* T **faça**
  $V_{tr_t} \leftarrow$ Remover os vértices cv_i do conjunto V_{tr}.
  $G_{tr_t} \leftarrow$ criarGrafo(G_{tr},V_{tr_t}).
  $h_t \leftarrow$ Aplicar o algoritmo de aprendizado (L) ao grafo (G_{tr_t}).
  Armazenar o modelo h_t.
fim
retorna $h_1, h_2, ..., h_T$

Algoritmo 4.6: Classificador CVCG - Etapa de Aprendizagem.

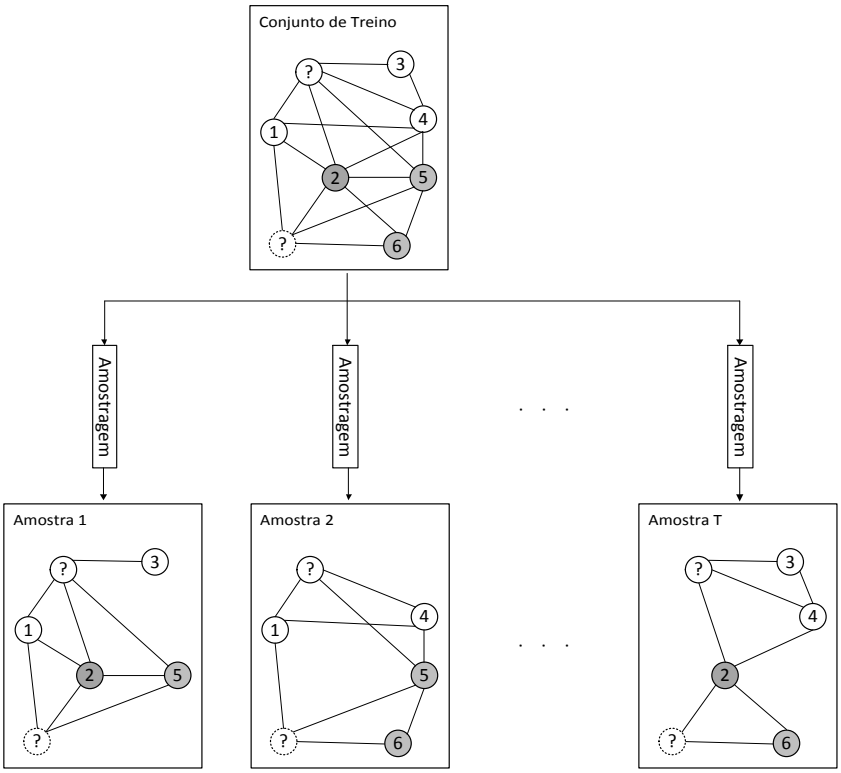


Figura 4.15: Amostragem de vértices no algoritmo CVCG.

CAPÍTULO

5

Avaliação Experimental

5.1 Metodologia

As avaliações foram realizadas com dados originalmente no formato proposicional, representados por tabelas atributo-valor, e em dados relacionais, representados no formato de grafo. Nestas avaliações, foram utilizados gráficos *boxplot* (Benjamini, 1988) para uma análise gráfica e o método de Demšar (2006) para determinar diferenças estatísticas entre os algoritmos utilizados.

5.1.1 Dados Proposicionais

Para os dados proposicionais foi empregado o esquema de *10-fold stratified cross-validation*, e foram utilizadas 3 versões dos dados: o conjunto original, o conjunto com 5% e com 10% de ruído. Nos experimentos, foram comparados os algoritmos *K-nearest Neighbor* (k=1, 3, 5, 7, 9, 15, 20, 30 e 40), a árvore de decisão C4.5 (Quinlan, 1993), o Naive Bayes (John e Langley, 1995), *Support Vector Machines* (SVM) usando o algoritmo *Sequential Minimum Optimization* (SMO) (Platt, 1999) e o algoritmo proposto A-KNN.

Estes algoritmos foram comparados com o algoritmo A-KNN em duas situações. Na primeira, a comparação foi somente com estes algoritmos isolados e na segunda, estes algoritmos foram utilizados como classificadores bases em um processo de *Boosting*. Para realizar as comparações foi empregado o método de Demšar (2006), o qual utiliza o teste de Friedman (1937, 1940) para

comparar vários algoritmos de uma só vez. A seguir, o A-KNN foi contrastado com o AdaNN (Sun e Huang, 2010), devido sua estratégia similar a um KNN adaptativo, isto é, dependendo do exemplo a classificar e sua vizinhança no conjunto de treino utiliza-se um determinado número K de vizinhos.

Os conjuntos proposicionais utilizados foram numéricos, sem valores ausentes e comumente utilizados em tarefas de classificação. Esses conjuntos foram obtidos do repositório UCI (Frank e Asuncion, 2010). A seguir, são descritos estes conjuntos e suas informações.

- Balance: Equilíbrio de peso.
- Blood: Transfusões de sangue.
- Breast_tissue: Medidas de impedância elétrica de amostras de tecido fresco retirado da mama.
- Ecoli: Classificação de sítios de localizações de proteínas.
- Glass: Amostras de vidro.
- Hayes-roth: Estudo de seres humanos.
- Heart-Statlog: Doença cardíaca.
- Ionosphere: Classificação de elétrons livres na ionosfera.
- Iris: Medições da sépalas e pétalas da planta Iris.
- Libras: LIBRAS (língua brasileira de sinais).
- Pima_diabetes: Doenças digestivas, do rim e diabetes.
- Segment: Segmentação de imagens.
- Sonar: Classificação de sinais sonoros.
- Statlog: Imagens de satélite.
- Teaching: Avaliações de desempenho docente.
- Vehicle: Silhuetas de veículos.
- Vowel: Reconhecimento de vogais.
- Wine: Vinhos cultivados numa mesma região de Itália.

- Wine-q-red: Qualidade do vinho.
- Wisconsin-breast-cancer: Classificação de câncer de mama.
- Yeast: Classificação de sítios de localizações de proteínas.
- Zoo: Identificação da classe de animais.

A Tabela 5.1 resume as informações, como a quantidade de exemplos, atributos e classes, de cada conjunto de dados.

Tabela 5.1: Conjuntos de Dados Numéricos.

Conjunto de Dados	**# Exemplos**	**# Atributos**	**# Classes**
balance (ba)	625	4	3
blood (bl)	748	4	2
breast_tissue (bt)	106	9	6
ecoli (ec)	336	7	8
glass (ga)	214	9	6
Hayes-Roth (hr)	132	5	3
heart-statlog (hs)	270	13	2
ionosphere (io)	351	34	2
iris (ir)	150	4	3
libras (li)	360	90	15
pima_diabetes (pd)	768	8	2
segment (se)	2310	19	7
sonar (so)	208	60	2
statlog (st)	94	18	4
teaching (te)	151	5	3
vehicle (ve)	846	18	4
vowel (vo)	990	11	11
wine (wi)	178	13	3
wine-q-red (wqr)	1599	11	6
w-breast-cancer (wbc)	699	9	2
yeast (ye)	1484	8	10
zoo (zo)	101	16	7

5.1.2 Dados Relacionais

Para dados representados em grafos, $G = (V, A)$, considerou-se o subconjunto de vértices com classe conhecida V^K (conjunto de treino), considerando uma amostra estratificada dos exemplos V. A seguir, o conjunto de teste (V^U) foi definido como $V - V^K$. Embora fosse desejável manter os dados de teste separados do treino, isto não é aplicável para dados em formato grafo

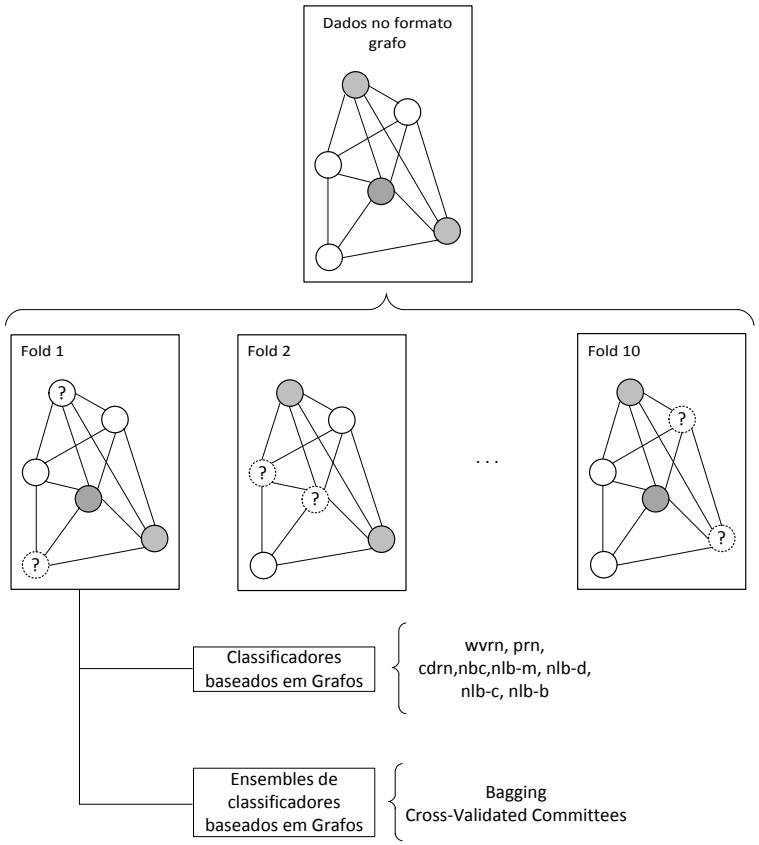


Figura 5.1: Esquema da Metodologia para dados no formato grafo. As cores no grafo representam as classes e os exemplos de teste são representados pelo símbolo "?".

(Macskassy e Provost, 2007). Nos experimentos, os conjuntos de treino e teste seguem o esquema de *10-fold stratified cross-validation*. Um exemplo desta representação é apresentada na Figura 5.1, onde o conjunto de treino V^K (vértices classificados -cinza e brancos- em cada *fold*) e o conjunto de teste V^U (vértices com o símbolo "?") são mantidos juntos nos grafos para o processo de validação cruzada.

A seguir, para cada *fold* foram avaliados os classificadores baseados em grafos: wvrn, prn, cdrn, nbc, nlb-m, nlb-d, nlb-c e nlb-b. Em seguida, para cada um destes classificadores foram empregadas as técnicas de *Bagging* e *Cross-Validated Committees*. Finalmente, foram comparadas as versões dos classificadores isolados e os *ensembles* usando o teste estatístico de Wilcoxon (1945), para verificar se o *ensemble* melhora a versão isolada. Foi utilizado o teste de Wilcoxon (1945) devido ao fato que compara dois classificadores, neste caso o classificador base e o classificador *ensemble*.

Já em relação aos conjuntos utilizados, estes foram obtidos diretamente no formato grafo. Os conjuntos de dados *adjnoun*, *football*, *polblogs*, *polbooks* foram obtidos do *site* do pesquisador Mark Newman[1]. Os demais conjuntos de dados foram obtidos do *site* da plataforma netkit[2]. A seguir, apresenta-se uma descrição destes conjuntos:

- adjnoun: grafo de adjacência de adjetivos e substantivos comuns no romance David Copperfield de Charles Dickens (Newman, 2006).

- fooball: grafo de jogos de futebol americano entre equipes colegiais durante a temporada do 2000. Os vértices são os equipes e as arestas ligam equipes que se enfrentaram. Este conjunto foi compilado por Girvan e Newman (2002).

- imdb-all: grafo de filmes na Internet. Os filmes são ligados se compartilham uma empresa de produção, produtor, diretor ou ator. Este conjunto de dados compilado por (Macskassy e Provost, 2007).

- imdb-prodco: grafo de filmes na Internet. Os filmes são ligados se compartilham uma empresa de produção. Este conjunto de dados foi compilado por (Macskassy e Provost, 2007).

- industry-yh: grafo de indústrias relacionadas por co-ocorrência em notícias obtidas na Web entre 04/01/1999 e 08/04/1999. Este grafo foi compilado por (Fawcett e Provost, 1999).

- industry-pr: grafo de indústrias relacionadas por co-ocorrência em notícias obtidas na Web entre 03/04/2003 e 30/09/2003. Esta grafo foi compilado por (Macskassy e Provost, 2007).

- polblogs: grafo de citações entre blogs políticos nos EUA, registrados em 2005 (Adamic e Glance, 2005).

- polbooks: grafo de livros políticos publicados no período de eleições de 2004 nos EUA. As arestas representam livros vendidos para um mesmo comprador. Estes dados foram compilados por Valid Krebs [1].

[1] http://www-personal.umich.edu/~mejn/netdata/
[2] http://netkit-srl.sourceforge.net/
[1] http://www.orgnet.com/

- webkb-cornell-cocite: grafo de páginas *Web* baseado no projeto WebKB (Craven et al., 1998) do departamento de ciência da computação da Universidade de Cornell. As páginas são ligadas se apresentam *hyperlinks* de co-citação (Se x liga y e y liga z, então x e y estão co-citando z).
- webkb-cornell-link: grafo de páginas *Web* baseado no projeto WebKB (Craven et al., 1998) do departamento de ciência da computação da Universidade de Cornell. As páginas são ligadas se compartilham algum *hyperlink*.
- webkb-texas-cocite: grafo de co-citação de páginas *Web* do projeto WebKB (Craven et al., 1998) para a Universidade de Texas.
- webkb-texas-link: grafo de *hyperlinks* em páginas *Web* do projeto WebKB (Craven et al., 1998) para a Universidade de Texas.
- webkb-washington-cocite: grafo de co-citação de páginas *Web* do projeto WebKB (Craven et al., 1998) para a Universidade de Washington.
- webkb-washington-link: grafo de *hyperlinks* em páginas *Web* do projeto WebKB (Craven et al., 1998) para a Universidade de Washington.
- webkb-wisconsin-cocite: grafo de co-citação de páginas *Web* do projeto WebKB (Craven et al., 1998) para a Universidade de Wisconsin.
- webkb-wisconsin-link: grafo de *hyperlinks* em páginas *Web* do projeto WebKB (Craven et al., 1998) para a Universidade de Wisconsin.

A seguir, apresenta-se a Tabela 5.2 com informações como a quantidade de vértices, arestas e classes de cada conjunto relacional.

5.2 Resultados

Nesta seção, são descritos os resultados obtidos na avaliação experimental.

5.2.1 Resultados com dados originalmente em formato Proposicional

A seguir são descritos os resultados obtidos em dados proposicionais, considerando 22 conjuntos de dados e 14 classificadores. Os resultados são resumidos nas Figuras 5.3, 5.4, 5.5, 5.7, 5.6 e 5.8 utilizando *boxplots* (Benjamini,

Tabela 5.2: Conjuntos de Dados Relacionais.

Conjunto de Dados	**# Vertices**	**# Arestas**	**# Classes**
adjnoun (adj)	112	850	2
football (foot)	115	1232	12
imdb-all (iall)	1441	51481	2
imdb-prodco (iprod)	1441	20317	2
industry-pr (ipr)	2189	13062	12
industry-yh (iyh)	1798	14165	12
polblogs (pblogs)	1490	19078	2
polbooks (pbooks)	105	882	3
WebKB-cornell-cocite (wcc)	351	26832	6
WebKB-cornell-link (wcl)	351	1393	6
WebKB-texas-cocite (wtc)	338	32988	6
WebKB-texas-link (wtl)	338	1002	6
WebKB-washington-cocite (wwac)	434	30463	6
WebKB-washington-link (wwal)	434	1941	6
WebKB-wisconsin-cocite (wwic)	354	33250	6
WebKB-wisconsin-link (wwil)	354	1155	6

1988) para facilitar a interpretação dos resultados e na Tabela 5.3 são mostradas as diferenças estatísticas. Tabelas com informação mais detalhada são apresentadas no Apêndice A.

Um gráfico *boxplot* mostrado na Figura 5.2 apresenta informações como a mediana ($q2$), o quartil inferior ($q1$), o quartil superior ($q3$), o limite inferior (li) e o limite superior (ls) dos dados analisados. Estes gráficos mostram uma tendência central dada pela mediana($q2$) e uma medida de dispersão dada pelo $IIQ = q3 - q1$ e os limites (li e ls).

Observa-se nas Figuras 5.3 e 5.4 que o método A-KNN apresentou uma mediana da acurácia maior que os outros classificadores, tanto na versão isolada (Figura 5.3) como na versão *Boosting* (Figura 5.4) dos algoritmos. Adicionalmente, sua dispersão foi comparável com ou menor do que as dos outros algoritmos. Além disso, seu limite inferior foi o segundo maior.

Para entender melhor o comportamento destes algoritmos, os dados foram separados em dois conjuntos: o primeiro com os dados originais e o segundo com dados com ruído. Em relação aos dados originais, conforme as Figuras 5.5 e 5.6, os algoritmos apresentaram resultados similares aos obtidos na análise usando tanto dados originais como com ruído (referidos como conjunto completo). Do mesmo modo, em relação aos resultados com dados somente com ruído, conforme as Figuras 5.7 e 5.8, também apresentaram resultados similares aos obtidos com o conjunto completo. Assim, o algoritmo A-KNN

Figura 5.2: Gráfico *boxplot* com informações da mediana ($q2$), quartil inferior ($q1$), quartil superior ($q3$), limite inferior (li) e limite superior (ls).

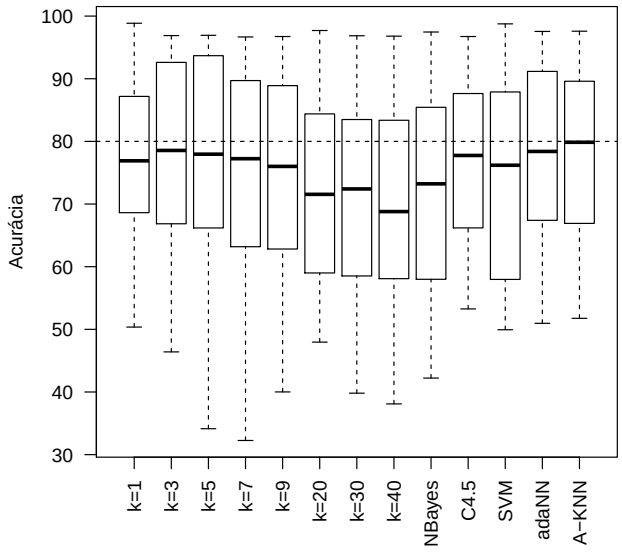


Figura 5.3: Resultados comparativos entre KNN, Naive Bayes, C4.5, SVM, AdaNN e o A-KNN em 22 conjuntos de dados, cada um com dois níveis de ruído (5% e 10%). Os resultados são representados em um *boxplot*.

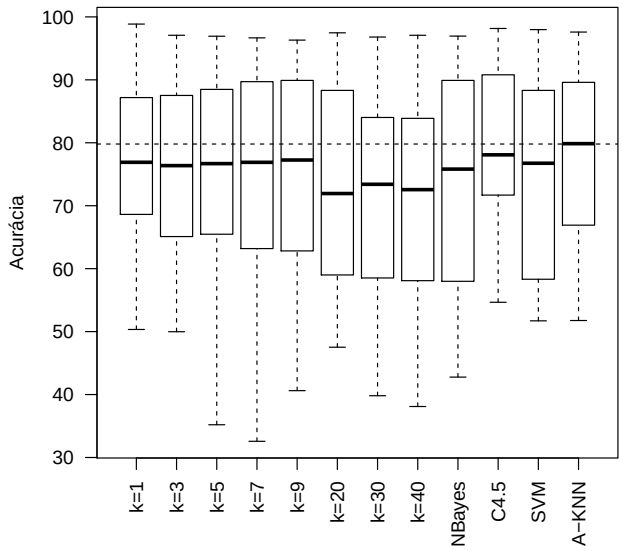


Figura 5.4: Resultados comparativos entre as versões *Boosting* do KNN, Naive Bayes, C4.5, SVM e o metodo proposto A-KNN em 22 conjuntos de dados, cada um com dois níveis de ruído (5% e 10%). Os resultados são representados em um *boxplot*.

apresentou bons resultados tanto nos dados originais como com ruído.

É interessante notar, como afirma Dietterich (2000b), que o *Boosting* parece ser especialmente suscetível a ruído. Ou seja, quando o número de *outliers* ou ruído é muito grande, o foco nesses exemplos difíceis pode tornar-se prejudicial para o desempenho do algoritmo. No experimento em dados com ruído, no entanto, a abordagem proposta mostrou um bom desempenho.

Em seguida, devido aos resultados semelhantes em dados originais e com ruído, foram utilizados os conjuntos de dados originais e com ruído juntos no estudo estatístico. Neste estudo, para determinar se os resultados apresentaram diferença estatística, foi empregado o método de Demšar (2006) usando o teste de Friedman (1937, 1940) com valor de confiança de $p = 0,05$. Para comparar pares de classificadores foi utilizado o pós-teste de Bonferroni-Dunn, onde $\triangle$ representa o ganho com diferença significativa para o A-KNN e - representa diferença não significativa entre par de classificadores.

O método de Demšar (2006) compara vários classificadores sobre vários conjuntos de dados. Este método atribui um *rank* para cada algoritmo em cada conjunto de dados considerando sua acurácia. O melhor algoritmo recebe o *rank* 1, o segundo 2 e assim por diante. No caso de empate, o *rank*

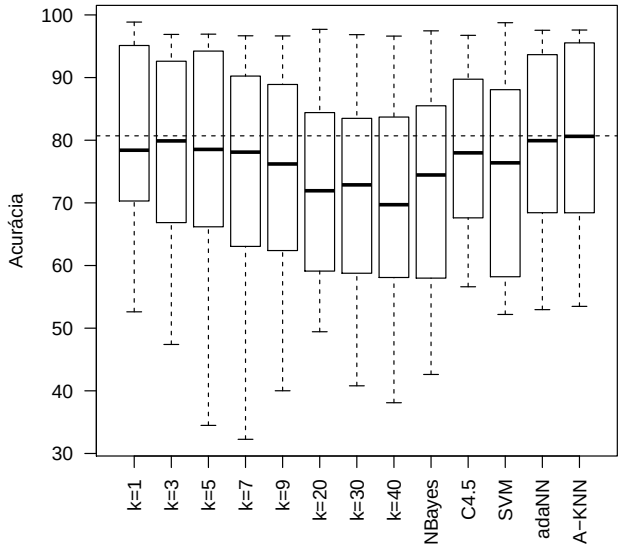


Figura 5.5: Resultados comparativos entre KNN, Naive Bayes, C4.5, SVM, AdaNN e o A-KNN em 22 conjuntos originais de dados. Os resultados são representados em um *boxplot*.

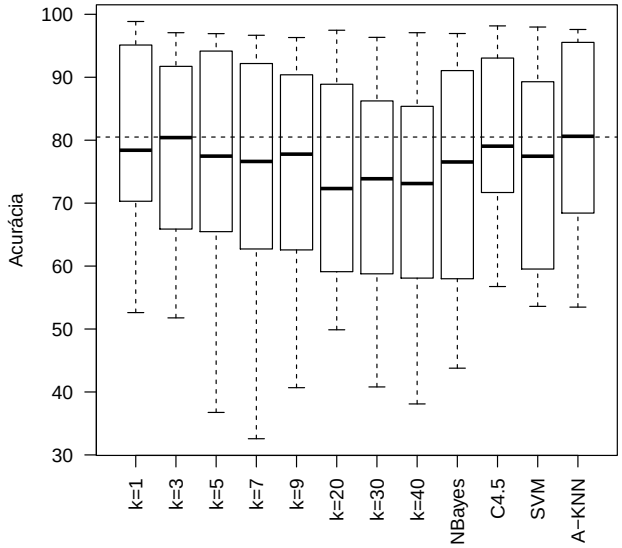


Figura 5.6: Resultados comparativos entre as versões *Boosting* do KNN, Naive Bayes, C4.5, SVM e o metodo proposto A-KNN em 22 conjuntos originais de dados. Os resultados são representados em um *boxplot*.

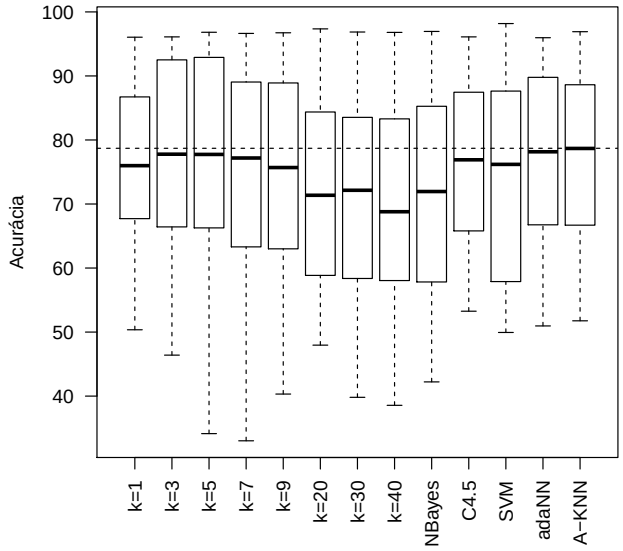


Figura 5.7: Resultados comparativos entre KNN, Naive Bayes, C4.5, SVM, AdaNN e o A-KNN em 22 conjuntos de dados com níveis de ruído de 5% e 10%. Os resultados são representados em um *boxplot*.

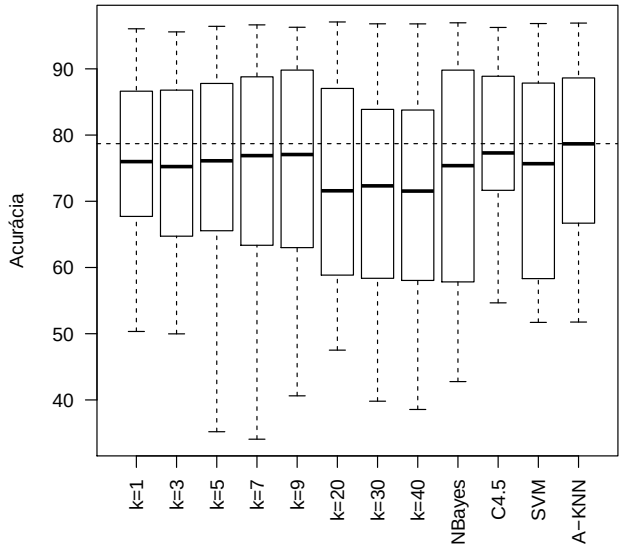


Figura 5.8: Resultados comparativos entre as versões *Boosting* do KNN, Naive Bayes, C4.5, SVM e o metodo proposto A-KNN em 22 conjuntos de dados com dois níveis de ruído (5% e 10%). Os resultados são representados em um *boxplot*.

médio é atribuído. Por exemplo, no caso de empate nas posições 3 e 4, o *rank* 3.5 é atribuído. A seguir, são calculadas as médias dos *ranks* para cada algoritmo utilizado no teste de Friedman. Este teste utiliza a distribuição F para calcular um valor crítico e rejeitar a hipótese nula considerando um nível de confiança. Considere N o número de conjunto de dados e K o número de classificadores testados.

No primeiro estudo de caso (comparando o A-KNN com KNN), $N = 66$ e $K = 10$, assim, o valor crítico para este estudo foi $cv \approx 1,8959$. Ou seja, qualquer resultado com o teste de Friedman acima deste valor crítico rejeitaria a hipótese nula. Neste experimento, o valor do teste de Friedman foi $F_f = 7,4733$, então a hipótese nula foi rejeitada, i.é, os algoritmos não são equivalentes.

A seguir utiliza-se um teste *post-hoc*, para comparar o desempenho do A-KNN com os outros algoritmos. Para tal, foi utilizado o teste de Bonferroni-Dunn. Este teste compara um algoritmo de controle (neste caso o A-KNN) com os outros algoritmos. Além disso, utiliza uma diferença crítica (CD) com um nível de confiança (p = 0,05). Qualquer diferença entre dois pares de algoritmos maior que CD, mostra que estes são significativamente diferentes. Neste experimento, a diferença critica foi $CD \approx 1,4615$. Finalmente, a Tabela 5.3 apresenta o estudo comparativo com o A-KNN.

Tabela 5.3: Comparação de algoritmos considerando o teste de Bonferroni-Dunn.

	sem *Boosting*											
KNN	k=1	k=3	k=5	k=7	k=9	k=15	k=20	k=30	k=40	NB	C4.5	SVM
A-KNN	-	△	-	-	-	-	-	△	△	△	-	-

	com *Boosting*											
KNN	k=1	k=3	k=5	k=7	k=9	k=15	k=20	k=30	k=40	NB	C4.5	SVM
A-KNN	-	△	△	△	△	-	-	△	△	-	-	-

Como segundo experimento, foi comparado o A-KNN com os algoritmos C4.5, SVM e Naive Bayes. A seguir, foram calculados os valores $N = 66$, $K = 4$, $CV \approx 2,6509$, $F_f = 4,1648$ e $CD \approx 0,538$. Portanto, este algoritmos não são equivalentes. Na Tabela 5.3 mostra-se as diferenças estatísticas.

Finalmente, foi considerada a versão *Boosting* dos algoritmos na comparação com o A-KNN. Na comparação com o KNN, foram calculados os valores $N = 66$, $K = 10$, $CV \approx 1,8959$, $F_f = 6,1228$ e $CD \approx 1,4615$. Conclui-se também que, os algoritmos não são equivalentes. Em seguida, a tabela 5.3 apresenta as diferenças estatísticas. Adicionalmente, o quarto experimento comparou as versões *Boosting* do Naive Bayes, C4.5 e SVM com o A-KNN. Neste experimento, foram calculados os valores $N = 66$, $K = 4$, $CV \approx 2,6509$, $F_f = 1,3962$

e $CD \approx 0,538$. Portanto, estes algoritmos são equivalentes (não existe uma diferença significativa entre eles).

Por outro lado, na Tabela A.3 foi comparado o A-KNN com outra abordagem de KNN adaptativo (*AdaNN*). A seguir, foi empregado o teste de Wilcoxon (devido a comparação de dois classificadores) para determinar se houve diferença estatística entre estes classificadores. Neste experimento, foi utilizado um valor de confiança de $\alpha = 0,05$ e os algoritmos não mostraram diferença estatística, portanto, estes algoritmos são equivalentes.

5.2.2 Resultados com dados Relacionais

Foram comparados classificadores baseados em grafos e suas versões *Bagging* com Duplicação de Arestas (BGDA), *Bagging* com Pesos (BGP), *Bagging* com Remoção de Vértices Duplicados (BGRVD) e *Cross-Validated Committees* (CVCG). As Figuras 5.9, 5.10, 5.11 e 5.12 apresentam os gráficos *boxplots* das comparações com as versões BGDA, BGP, BGRVD e CVCG, respectivamente. Adicionalmente, tabelas com informação mais detalhada são apresentadas no Apêndice B.

Para determinar a diferença estatística nos pares de algoritmos também foi empregado o método de Demšar (2006) e o teste de Wilcoxon (1945) com valor de confiança de $\alpha = 0,05$. A Tabela 5.4 resume esta comparação, onde $\triangle$ representa diferença significativa (o *ensemble* melhorou o algoritmo base) e - representa diferença não significativa entre um par de classificadores.

Tabela 5.4: Comparação estatística dos algoritmos baseados em grafos e suas versões *ensembles*: BGDA, BGRVD, BGP e CVCG. O teste empregado foi o de Wilcoxon.

	BGDA	BGRVD	BGP	CVCG
wvrn	-	-	-	-
prn	$\triangle$	$\triangle$	$\triangle$	$\triangle$
cdrn	-	-	-	$\triangle$
nbc	-	-	-	-
nlb-m	-	-	-	-
nlb-d	-	-	-	-
nlb-c	-	-	-	-
nlb-b	$\triangle$	$\triangle$	$\triangle$	-

Em primeiro lugar, com relação à versão *Bagging* com Duplicação de Arestas e a Figura 5.9, o classificador *BGDAprn* apresentou uma mediana maior que *prn*. Além disso, a mediana de *BGDAnlb-b* foi um pouco maior que o

nlb-b e sua dispersão foi menor (seu limite inferior é maior). Por outro lado, o *BGDAnlb-m* apresentou uma menor mediana, no entanto seu limite inferior foi maior. Do mesmo modo, o *BGDAcdrn* apresentou um limite inferior maior que o *cdrn*. Finalmente, os outros algoritmos não apresentaram grandes diferenças entre suas versões. Por conseguinte, utilizando o teste de Wilcoxon pode-se ver na Tabela 5.4 que o *ensemble* melhorou significativamente os algoritmos *prn* e *nlb-b*.

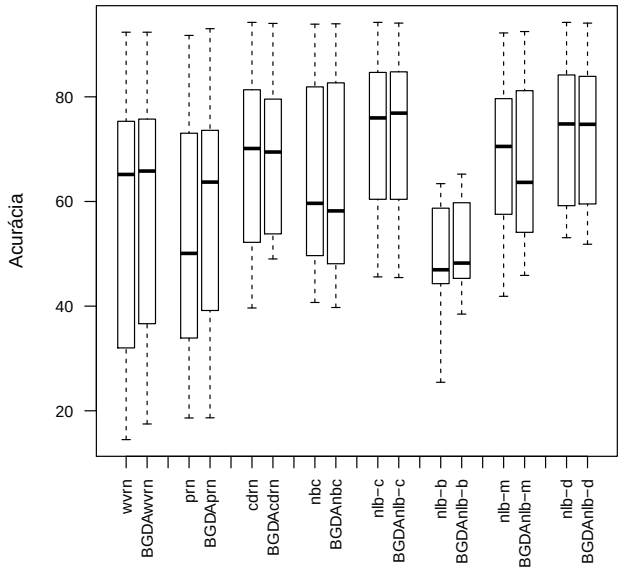


Figura 5.9: Resultados comparativos entre classificadores baseados em grafos (wvrn, prn, cdrn, nbc, nlb-m, nlb-d, nlb-c e nlb-b) e o *ensemble* BGDA em 16 conjuntos de dados. Os resultados são representados em *boxplots*.

Em segundo lugar, com relação à versão *Bagging* com Pesos (Figura 5.10), os classificadores *prn*, *cdrn* e *nlb-b* apresentaram o mesmo comportamento na versão *Bagging*. Por outro lado, o *BGPnlb-m* e *nlb-m* apresentaram medianas parecidas, no entanto o limite inferior de *BGPnlb-m* foi maior. Finalmente, os outros algoritmos não apresentaram grandes diferenças entre suas versões. Por conseguinte, utilizando o teste de Wilcoxon, demonstra-se que o *ensemble* melhora significativamente os algoritmos *prn* e *nlb-b* (Tabela 5.4).

Em terceiro lugar, com relação à versão *Bagging* com Remoção de Vértices Duplicados (Figura 5.11), os resultados foram, semelhantes à versão *Bagging* com pesos. A seguir, foi aplicado o teste de Wilcoxon e observa-se que o *ensemble* melhorou significativamente os algoritmos *prn* e *nlb-b* (Tabela 5.4).

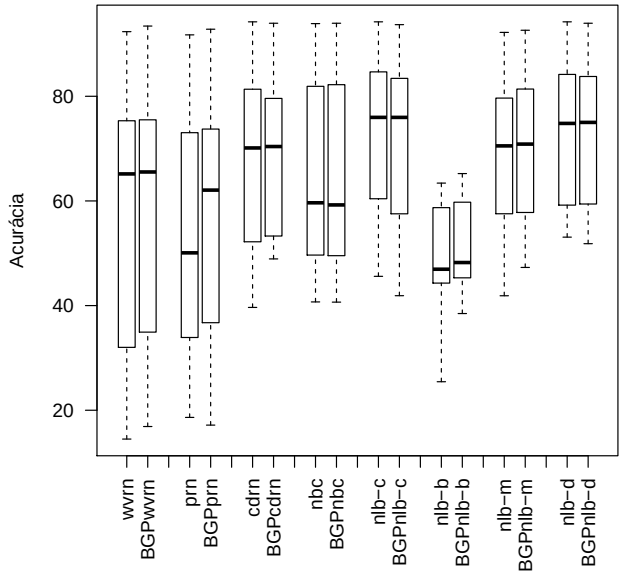


Figura 5.10: Resultados comparativos entre classificadores baseados em grafos (wvrn, prn, cdrn, nbc, nlb-m, nlb-d, nlb-c e nlb-b) e o *ensemble* BGP em 16 conjuntos de dados. Os resultados são representados em *boxplots*.

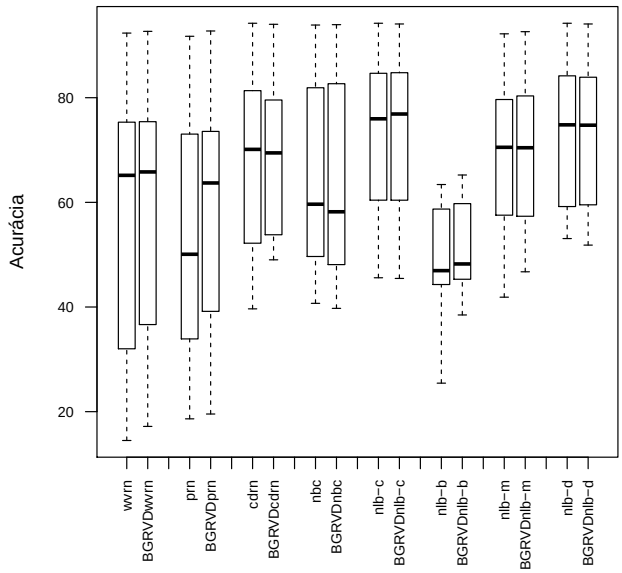


Figura 5.11: Resultados comparativos entre classificadores baseados em grafos (wvrn, prn, cdrn, nbc, nlb-m, nlb-d, nlb-c e nlb-b) e o *ensemble* BGRVD em 16 conjuntos de dados. Os resultados são representados em *boxplots*.

Com relação à versão *Cross-Validated Committees* (Figura 5.12), o classificador *CVCGprn* apresentou uma mediana maior que *prn*. Por outro lado, os classificadores *cdrn*, *nlb-m* e *nlb-b* apresentaram medianas semelhantes com sua versão *Cross-Validated Committees*, porém os limites inferiores da sua versão *ensemble* foram maiores. Além disso, os outros classificadores não apresentaram grandes diferenças entre suas versões. Finalmente, aplicando o teste de Wilcoxon observa-se que o *ensemble* melhorou significativamente os algoritmos *prn* e *cdrn* (Tabela 5.4).

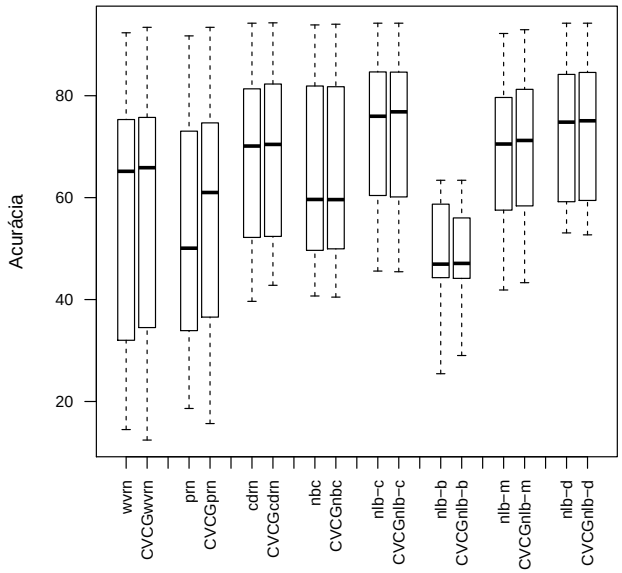


Figura 5.12: Resultados comparativos entre classificadores baseados em grafos (wvrn, prn, cdrn, nbc, nlb-m, nlb-d, nlb-c e nlb-b) e o *ensemble* CVCG em 16 conjuntos de dados. Os resultados são representados em *boxplots*.

CAPÍTULO

6

Discussão: O Papel da Diversidade nos *Ensembles*

Para analisar o comportamento dos métodos *ensembles* propostos, foram empregados diagramas k-erro (Margineantu e Dietterich, 1997). Um diagrama k-erro consiste de uma medida estatística kappa (k) (Cohen, 1960; Agresti, 2002), para representar a concordância entre classificadores, e uma medida de $erro$, para representar a média das taxas de erro destes classificadores. Estes diagramas ajudam a interpretar a relação entre acurácia e diversidade de cada classificador do *ensemble* (Dietterich, 2000b).

Para cada par de classificadores no *ensemble*, a acurácia é representada pela média das taxas de erro ($erro$) em dados de teste, e a diversidade é representada por uma medida de concordância ($kappa$) entre estes dois classificadores. Esta medida pode obter valores: (a) quando $kappa = 0$, a concordância dos dois classificadores é igual ao esperado por acaso; (b) quando $kappa = 1$, os dois classificadores estão de acordo em todos os exemplo; (c) quando $kappa < 0$, existe desacordo entre os classificadores e (d) quando $kappa$ está no intervalo $[0, 1]$, indica o grau de concordância entre os classificadores.

Nos experimentos feitos para cada *fold* no *ensemble* foram considerados 10 classificadores. Estes 10 classificadores são combinados de 2 em 2 gerando $C_2^{10} = 45$ pontos por cada *fold* no diagrama k-erro. Finalmente, dado que são 10 *folds* no total foram gerados $45 * 10 = 450$ pontos no diagrama k-erro para cada conjunto de dados.

A seguir, estes diagramas foram empregados para estudar as técnicas de *Bagging* baseado em grafos e *Cross- Validated Committees* baseados em grafos.

6.1 *Bagging* baseado em grafos

Primeiramente, para analisar a influência da diversidade entre os classificadores no *Bagging* baseado em Grafos com Duplicação de Arestas (BGDA), foram consideradas algumas configurações (conjunto de dados e classificadores) nas quais a técnica substancialmente melhora, diminui ou mantém a acurácia do classificador base. Esses casos foram escolhidos para procurar compreender os fatores preponderantes para o sucesso do *ensemble*.

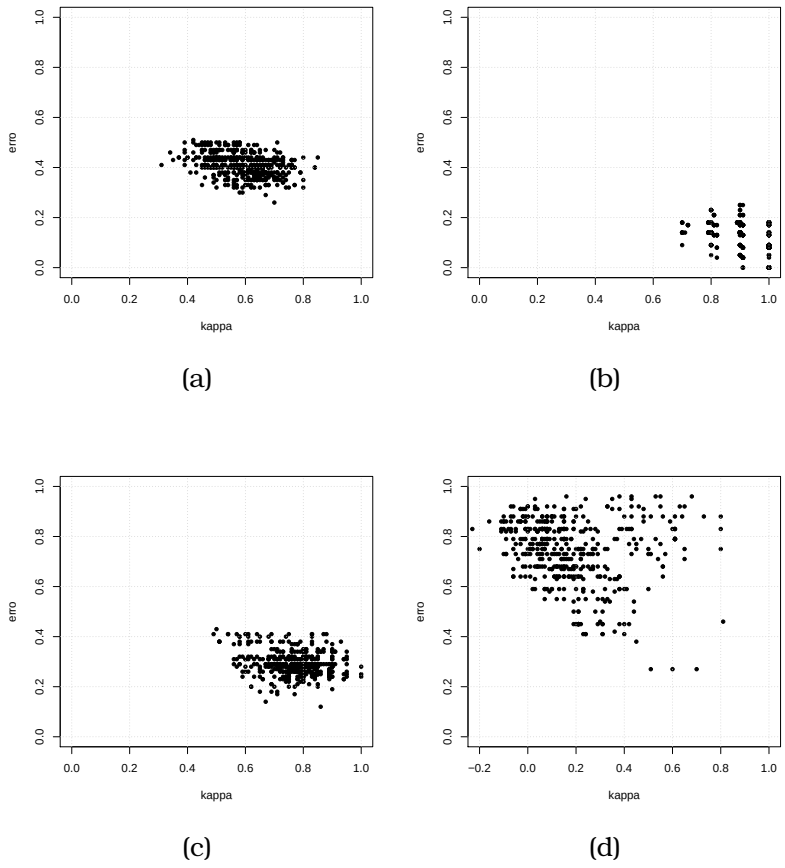


Figura 6.1: Casos de Estudo do BGDA. (a) $WebKB - texas - cocite$ usando prn (ganho substancial), (b) $football$ usando prn (pequena redução), (c) $WebKB - texas - cocite$ usando $nlb - m$ (redução substancial), e (d) $football$ usando $nlb - b$ (ganho substancial)

No primeiro caso de estudo apresentado na Figura 6.1(a), o *ensemble* me-

lhora substancialmente no conjunto de dados $WebKB - texas - cocite$ com o classificador base prn. Neste caso, a taxa de erro do classificador base é $ecb = 53,17\%$ e as taxas individuais dos classificadores do *ensemble* são menores que ecb. Adicionalmente, a maioria destes classificadores (66%) tem um valor de $kappa$ no intervalo $[0,5; 0,7]$. Estes resultados indicam que taxas de erros do *ensemble* menores do que as do o classificador base e uma diversidade razoável entre os classificadores do *ensemble* melhora a acurácia do *ensemble*.

No segundo caso de estudo apresentado na Figura 6.1(b), o *ensemble* diminui um pouco a acurácia no conjunto de dados $football$ com o classificador base prn. Neste caso, a taxa de erro do classificador base é $ecb = 8,71\%$ e $68,89\%$ dos classificadores individuais do *ensemble* apresentam uma taxa de erro maior que ecb. Além disso, $90,22\%$ destes classificadores mostram um $kappa$ no intervalo $[0,8; 1]$. Estes dados indicam que taxas de erro de classificadores no *ensemble* maiores que a taxa do classificador base, com menor diversidade não melhora a acurácia no *ensemble*.

No terceiro caso de estudo apresentado na Figura 6.1(c), o *ensemble* diminui substancialmente a acurácia no conjunto de dados $WebKB - texas - cocite$ com o classificador base $nlb - m$. Neste caso, a taxa de erro do classificador base é $ecb = 27,18\%$ e as taxas dos classificadores individuais no *ensemble* são maiores que ecb em $69,78\%$. Do mesmo modo, foi calculado que $72,98\%$ dos classificadores apresentam $kappa$ no intervalo $[0,7; 0,9]$. Estes dados indicam que taxas de erro nos classificadores no *ensemble* maiores que as do classificador base e pouca diversidade decrementa substancialmente a acurácia do *ensemble*.

No último caso de estudo mostrado na Figura 6.1(d), o *ensemble* substancialmente melhora a acurácia no conjunto de dados $football$ com o classificador base $nlb - b$. Neste caso, a taxa de erro do classificador base é $ecb = 74,55\%$ e $55,77\%$ das taxas do erro dos classificadores do *ensemble* são maiores que ecb. Adicionalmente, foi calculado que $60,22\%$ destes classificadores mostram um $kappa$ no intervalo $[0; 0,3]$. Estes dados indicam que, ainda com classificadores com taxas de erro maiores que o erro do classificador base, o *ensemble* melhora significativamente a acurácia. Este resultado mostra a relevância da diversidade em *ensembles*.

Posteriormente, considerando o algoritmo de *Bagging* baseado em Grafos com Remoção de Vértices Duplicados (BGRVD). Foram criados os diagramas k-erro, apresentados na Figura 6.2, para os mesmos casos de estudo, os casos de $WebKB - texas - cocite$ usando prn, $football$ usando prn e $football$ usando

$nlb - b$ apresentaram resultados semelhantes ao algoritmo de *Bagging* baseado em grafos. Já no caso de $WebKB - texas - cocite$ usando $nlb - m$, Figura 6.2(c), as taxas dos classificadores individuais com erro maior que o classificador base diminuem a $66,44\%$. Do mesmo modo, é calculado que $89,55\%$ dos classificadores apresentaram $kappa$ no intervalo $[0,6; 0,9]$, indicando um maior porcentagem na faixa de $[0,6; 0,8]$. Esta diversidade ajuda a melhorar a acurácia neste *ensemble*, em relação ao BGDA.

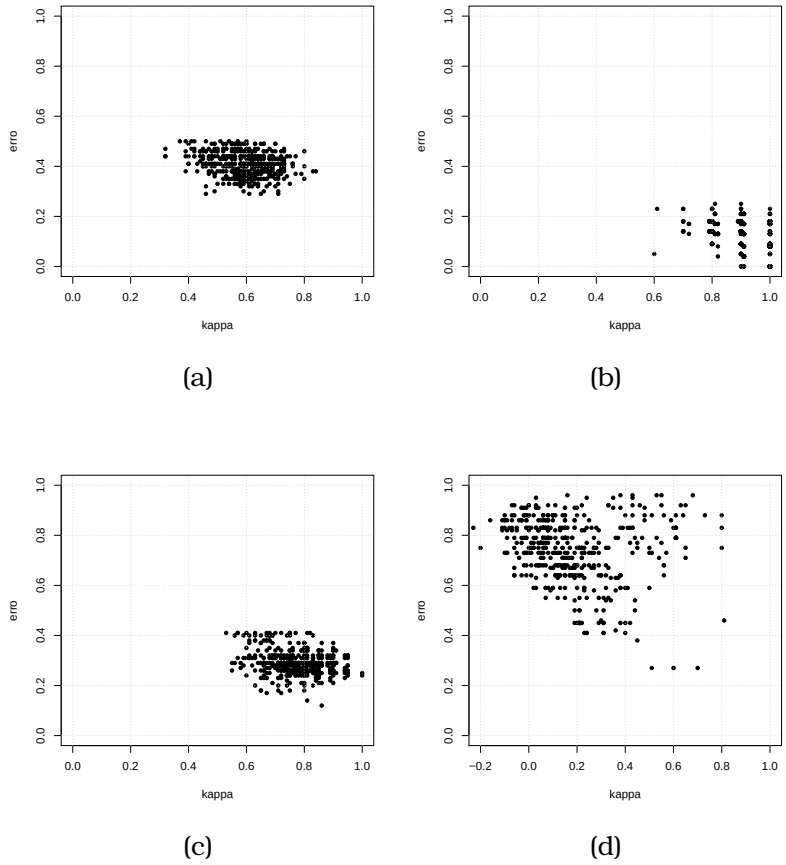


Figura 6.2: Casos de Estudo do BGRVD. (a) $WebKB - texas - cocite$ usando prn (ganho substancial), (b) $football$ usando prn (pequena redução), (c) $WebKB - texas - cocite$ usando $nlb - m$ (pequena redução), e (d) $football$ usando $nlb - b$ (ganho substancial)

Finalmente, considerando o algoritmo de *Bagging* baseado em Pesos (BGP), na Figura 6.3 são apresentados seus diagramas k-erro. Neste algoritmo, os casos de $football$ usando prn e $football$ usando $nlb - b$ apresentaram resultados semelhantes com os outros algoritmos de *Bagging*. No caso de $WebKB - texas - cocite$ usando prn, apresentado na Figura 6.3(a), o *ensemble* melhorou o classi-

ficador base. No entanto, as taxas individuais dos classificadores no *ensemble* são menores que o classificador base em um $55,56\%$ e $64,67\%$ destes classificadores tem um valor de $kappa$ no intervalo $[0,4; 0,5]$. Neste caso, é importante a diversidade dos classificadores para a melhoria da acurácia. Em seguida, no caso de $WebKB - texas - cocite$ usando $nlb - m$ apresentado na Figura 6.3(c), o *ensemble* apresenta uma melhoria na acurácia em relação ao BGDA. Esta melhoria, é devido ao fato que $64,44\%$ dos classificadores individuais do *ensemble* apresentaram um $kappa$ no intervalo $[0,6; 0,8]$. Estes resultados novamente confirmam a importancia da diversidade em *ensembles*.

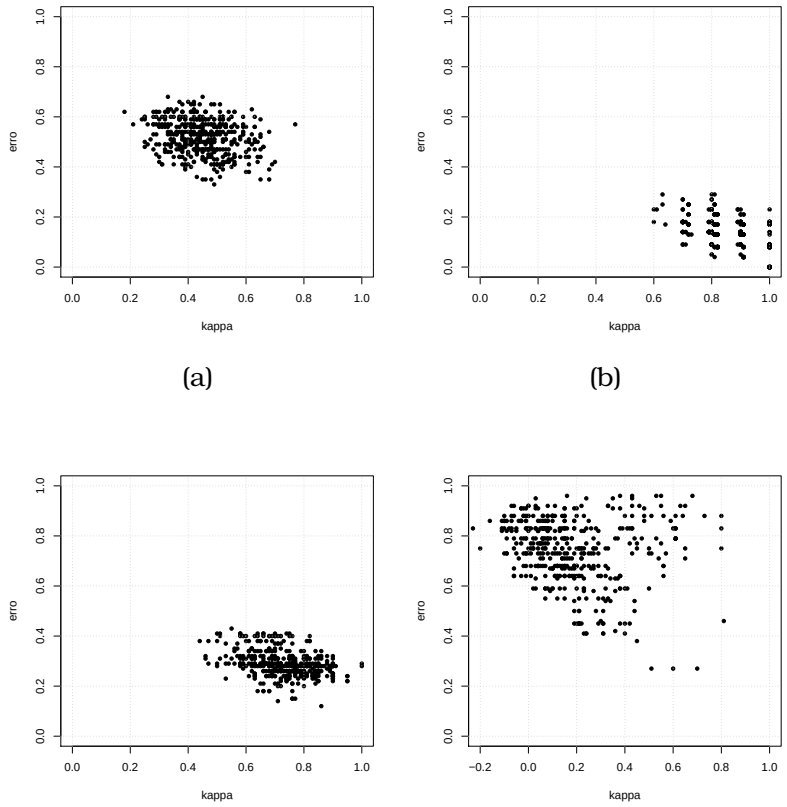


Figura 6.3: Casos de Estudo do BGP. (a) $WebKB - texas - cocite$ usando prn (ganho substancial), (b) $football$ usando prn (pequena redução), (c) $WebKB - texas - cocite$ usando $nlb - m$ (pequena redução), e (d) $football$ usando $nlb - b$ (ganho substancial)

6.2 *Cross-Validated Committees* baseados em grafos

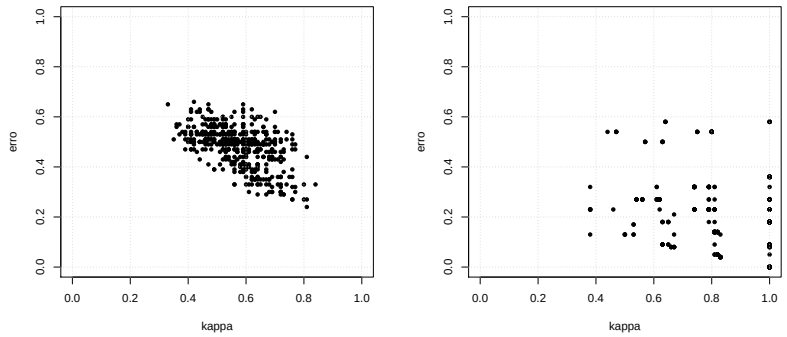
Do mesmo modo que o *Bagging* baseado em grafos, no *Cross-Validated Committees* baseado em Grafos (CVCG), foram considerados alguns conjuntos de dados e classificadores onde a técnica substancialmente melhora, diminui ou mantém a acurácia do classificador base.

No primeiro caso de estudo apresentado na Figura 6.4(a), o *ensemble* melhora substancialmente no conjunto de dados $WebKB - texas - cocite$ com o classificador base prn. Neste caso, a taxa de erro do classificador base é $ecb = 53,17\%$ e $75,11\%$ dos classificadores individuais do *ensemble* apresentam uma taxa de erro menor que ecb. Adicionalmente, a maioria destes classificadores ($62,66\%$) tem um valor de $kappa$ no intervalo $[0,5; 0,7]$. Estes resultados confirmam que taxas menores que o classificador base e uma diversidade razoável melhora a acurácia do *ensemble*.

No segundo caso de estudo apresentado na Figura 6.4(b), o *ensemble* melhora substancialmente no conjunto de dados $adjnoun$ com o classificador base $cdrn$. Neste caso, a taxa de erro do classificador base é $ecb = 19,47\%$ e $58,00\%$ dos classificadores individuais do *ensemble* apresentam uma taxa de erro menor que ecb. Adicionalmente, a maioria destes classificadores ($67,56\%$) tem um valor de $kappa$ no intervalo $[0,8; 1]$. Estes dados indicam que taxas de erro de classificadores no *ensemble* menores que a taxa do classificador base, mesmo com pouca diversidade melhora a acurácia no *ensemble*.

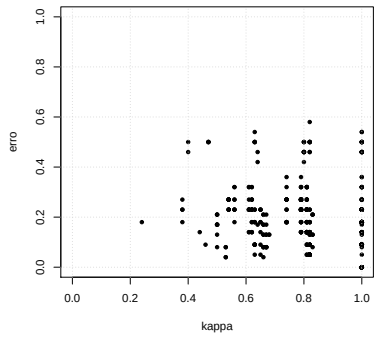
No terceiro caso de estudo apresentado na Figura 6.4(c), o *ensemble* melhora substancialmente no conjunto de dados $adjnoun$ com o classificador base $nlb-m$. Neste caso, a taxa de erro do classificador base é $ecb = 18,56\%$ e $57,78\%$ dos classificadores individuais do *ensemble* apresentam uma taxa de erro menor que ecb. Adicionalmente, a maioria destes classificadores ($61,78\%$) tem um valor de $kappa$ no intervalo $[0,8; 1]$. Estes dados confirmam que taxas de erro de classificadores no *ensemble* menores que a taxa do classificador base, mesmo com pouca diversidade melhora a acurácia no *ensemble*.

No quarto caso de estudo apresentado na Figura 6.4(d), o *ensemble* mantém a acurácia no conjunto de dados $adjnoun$ com o classificador base nbc. Neste caso, a taxa de erro do classificador base é $ecb = 17,80\%$ e $55,77\%$ dos classificadores individuais do *ensemble* apresentam uma taxa de erro maior que ecb. Além disso, $85,77\%$ destes classificadores mostram um $kappa$ no intervalo $[0,7; 1]$. Estes dados confirmam que taxas de erro de classificadores no *ensemble* maiores que a taxa do classificador base com menor diversidade não

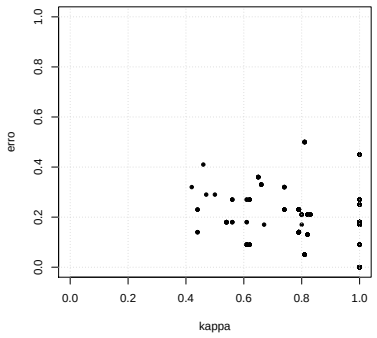


(a)

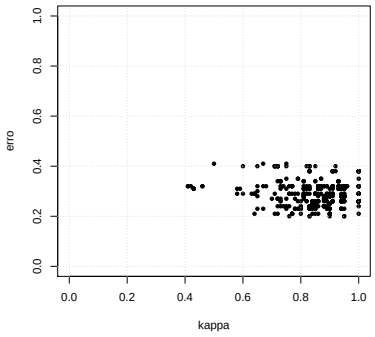
(b)



(c)



(d)



(e)

Figura 6.4: Casos de Estudo do CVCG. (a) $WebKB - texas - cocite$ usando prn (ganho substancial), (b) $adjnoun$ usando $cdrn$ (ganho substancial), (c) $adjnoun$ usando $nlb - m$ (ganho substancial), (d) $adjnoun$ usando nbc (empate) e (e) $WebKB - texas - cocite$ usando $cdrn$ (pequena redução).

melhora a acurácia no *ensemble* (neste caso a acurácia foi mantida).

No quinto caso de estudo apresentado na Figura 6.4(e), o *ensemble* diminui um pouco a acurácia no conjunto de dados $WebKB - texas - cocite$ com o classificador base $cdrn$. Neste caso, a taxa de erro do classificador base é $ecb = 27,76\%$ e $64,44\%$ dos classificadores individuais do *ensemble* apresentam uma taxa de erro maior que ecb. Além disso, $79,11\%$ destes classificadores mostram um $kappa$ no intervalo $[0,8; 1]$. Estes dados indicam que taxas de erro de classificadores no *ensemble* maiores que a taxa do classificador base e com pouca diversidade não melhora a acurácia no *ensemble* (neste caso a acurácia diminuiu um pouco). Observe que neste caso, a percentagem dos classificadores individuais com maior erro aumentou em relação ao quarto caso e apresenta um nível maior de concordância (maior valor de $kappa$).

Finalmente, os casos de estudo deste capítulo são resumidos na Tabela 6.1. Nesta tabela é importante ressaltar que um nível baixo ou médio de $kappa$ (alta ou média diversidade, respectivamente) ajuda na melhoria do desempenho do *ensemble*. Por outro lado, um nível alto de $kappa$ não melhora o desempenho do *ensemble* lembrando que o $kappa$ alto significa pouca diversidade entre os classificadores. Além disso, em relação aos erros no classificador base (ecb) e os erros no *ensemble* (ee), é importante notar que quando o $ee < ecb$, o *ensemble* melhora o desempenho do classificador base.

Tabela 6.1: Tabela comparativa dos casos de estudo. Onde $\uparrow$, $-$ e $\downarrow$ representam melhoria, empate ou perda no desempenho do *ensemble*. Além disso, ecb é o erro do classificador base e ee é o erro dos classificadores do *ensemble*.

	$kappa$		
	baixo	médio	alto
$ee < ecb$		$4(\uparrow)$	$2(\uparrow)$
$ee > ecb$	$3(\uparrow)$		$8(-/\downarrow)$

CAPÍTULO

7

Conclusões

A grande maioria dos algoritmos *ensembles* utiliza dados estruturados em uma representação proposicional para construção de modelos computacionais. Tal representação limita-se a descrever características individuais dos objetos representados, sem considerar relações existentes entre eles. Porém, existem domínios em que os dados são naturalmente representados em um formato relacional, como em redes de co-autoria, redes de ligações telefônicas, entre outros.

Neste trabalho foram propostos algoritmos *ensembles* para classificadores relacionais baseados em grafos, que contribuem para suprir a lacuna observada na literatura. Em nossa investigação não foram encontrados estudos sobre *ensembles* no contexto de classificadores relacionais baseados em grafos. Demonstrou-se que estes *ensembles* melhoram ou mantém o desempenho do seus classificadores relacionais bases. Além disso, três aspectos desta pesquisa devem ser destacados. O primeiro, foi em relação ao estudo do algoritmo *Boosting* relacional que originou o algoritmo A-KNN. O segundo, foi na adaptação da etapa de amostragem nos classificadores *Bagging* e *Cross-Validated Committees* para dados no formato de grafo. E o terceiro, foi no estudo da importância da diversidade em algoritmos *ensembles* baseados em grafos.

Em relação ao classificador A-KNN, a avaliação foi dividida em: avaliação com outros algoritmos de aprendizado, avaliação com versões *Boosting* e finalmente, uma avaliação com o algoritmo AdaNN (devido ao seu comportamento

similar com o A-KNN). Nas duas primeiras avaliações, o algoritmo proposto mostrou melhoria estatísticamente significativa ou empate em relação aos outros algoritmos. E na terceira comparação, mostrou resultados equivalentes.

Em relação aos classificadores *Bagging* e *Cross-Validated Committees* baseados em Grafos, a avaliação foi dividida em avaliação destes classificadores com seus classificadores bases e estudo do papel da diversidade entre classificadores que formam o *ensemble*. Em primeiro lugar, comparando as três versões do algoritmo de *Bagging* e o *Cross-Validated Committees* com seus classificadores base, estes *ensembles* apresentaram melhores resultados. A seguir, foram criados diagramas k-erro para estudar a diversidade destes classificadores. O resultado deste estudo mostrou que um valor de $kappa$ baixo ou médio, isto é classificadores bases com alta ou média diversidade, melhoram o desempenho do *ensemble*.

A seguir, nesta seção são descritas as principais contribuições, limitações e trabalhos futuros.

7.1 Principais Contribuições

As principais contribuições deste trabalho foram:

1. desenvolvimento do algoritmo KNN Adaptativo baseado em Grafos, o qual apresentou uma melhoria significativa ou empate na comparação com outros algoritmos;

2. desenvolvimento de uma técnica de *Bagging* baseada em Grafos, a qual melhorou ou manteve o desempenho do seus classificadores bases;

3. desenvolvimento do *ensemble Cross-Validated Committees* baseado em Grafos, o qual melhorou ou manteve o desempenho do seus classificadores bases;

4. análise e importância da diversidade em classificadores *ensembles* baseados em grafos.

As técnicas comentadas deram origem as seguintes publicações: (Murrugarra-Llerena e de Andrade Lopes, 2011a) e (Murrugarra-Llerena e de Andrade Lopes, 2011b), já comentadas na introdução.

No desenvolvimento do algoritmo A-KNN, foram implementadas funções para entender o comportamento e evolução do algoritmo. Além disso, foram

adicionadas algumas novas características ao PEx (Paulovich et al., 2007) para visualização de grafos e análise visual do A-KNN.

Considerando os *ensembles Bagging* e *Cross-Validated Committees*, foram desenvolvidos estes algoritmos usando a ferramenta Netkit (Macskassy e Provost, 2007). Além disso, foi desenvolvida uma função para a geração dos gráficos k-erro para análise da diversidade nos classificadores *ensembles*.

7.2 Limitações

Este trabalho possui algumas limitações em relação a utilização das técnifecas propostas. Uma primeira limitação refere-se à atribuição do peso no algoritmo A-KNN, neste caso foi atribuído o valor 1. No entanto, podem ser atribuídos valores de similaridade nas arestas usando a distância euclidiana, manhattan, entre outras.

Outra limitação é em relação ao uso do classificador base, neste caso foi utilizado o algoritmo *wvrn*. Porém, existem outros classificadores baseados em grafos a serem explorados.

Além disso, todos os algoritmos aqui propostos possuem um alto custo de processamento, o que dificulta sua aplicação em conjuntos de dados com grande quantidade de exemplos. O alto custo de processamento é devido, a geração de vários classificadores individuais no processo de treino e classificação. Uma análise mais detalhada de complexidade dos algoritmos deve ser efetuada em trabalhos futuros. No entanto, estes algoritmos apresentam um melhor desempenho.

7.3 Trabalhos Futuros

Considerando o algoritmo A-KNN, um futuro trabalho será a exploração desta técnica em bases de dados com classes desbalanceadas. Devido ao fato que classes com poucos exemplos devam considerar um k menor e classes com vários exemplos podem explorar um valor k maior, este algoritmo poderá mostrar resultados relevantes neste domínio.

Em relação as técnicas de *Bagging* no domínio de grafos, será realizado um estudo com dados com ruído para contrastar os resultados de Dietterich (2000b) em dados representados em tabelas atributo-valor. Segundo Dietterich (2000b), ao contrario do *Boosting*, quando é adicionado ruído nos dados, o *Bagging* melhora ou mantém o desempenho do seu classificador base.

Finalmente, em relação as técnicas de *Bagging* e *Cross-Validated Committees* no domínio de grafos, serão exploradas outros *ensembles*. Podendo ser utilizados as técnicas de *Wagging*, *Bagging* usando Diversidade, *Trimmed Bagging*, *Stacking*, entre outras.

Referências Bibliográficas

Adamic, L. A. e Glance, N. (2005). The political blogosphere and the 2004 u.s. election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, LinkKDD '05, páginas 36–43, New York, NY, USA. ACM. Citado na página 65.

Agresti, A. (2002). *Categorical Data Analysis*. Wiley Series in Probability and Statistics. Wiley-Interscience, 2nd[a] edição. Citado na página 77.

Alpaydin, E. (2004). *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press. Citado na página 1.

Bauer, E. e Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Mach. Learn.*, 36:105–139. Citado na página 14.

Belkin, M. e Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15:1373–1396. Citado na página 4.

Benjamini, Y. (1988). Opening the box of a boxplot. *The American Statistician*, 42:257–262. Citado nas páginas 61 e 66.

Breiman, L. (1996). Bagging predictors. *Mach. Learn.*, 24(2):123–140. Citado nas páginas 14 e 16.

Bruzzone, L., Cossu, R., e Vernazza, G. (2004). Detection of land-cover transitions by combining multidate classifiers. *Pattern Recogn. Lett.*, 25:1491–1500. Citado na página 2.

Bryll, R. (2003). Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognition*, 36(6):1291–1302. Citado na página 18.

Chakrabarti, S., Dom, B., e Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. *SIGMOD Rec.*, 27(2):307–318. Citado nas páginas 5, 27, 28, 29, e 31.

Cherkauer, K. (1996). Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks. In *Working Notes of the AAAI Workshop on Integrating Multiple Learned Models*, páginas 15–21. Citado na página 15.

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46. Citado na página 77.

Cortes, C., Pregibon, D., e Volinsky, C. (2001). Communities of interest. In *In Proceedings of the Fourth International Conference on Advances in Intelligent Data Analysis (IDA)*, páginas 105–114. Citado na página 5.

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., e Slattery, S. (1998). Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI '98/IAAI '98, páginas 509–516, Menlo Park, CA, USA. American Association for Artificial Intelligence. Citado na página 66.

Croux, C., Joossens, K., e Lemmens, A. (2007). Trimmed bagging. *Computational Statistics & Data Analysis*, 52(1):362–368. Citado na página 18.

De Raedt, L. (2008). *Logical and Relational Learning: From ILP to MRDM (Cognitive Technologies)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA. Citado na página 3.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30. Citado nas páginas v, vii, 7, 40, 61, 69, e 73.

Dietterich, T. G. (2000a). Ensemble methods in machine learning. In *MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems*, páginas 1–15, London, UK. Springer-Verlag. Citado nas páginas xi, 9, 10, 11, e 35.

Dietterich, T. G. (2000b). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Mach. Learn.*, 40:139–157. Citado nas páginas 5, 6, 69, 77, e 87.

Dietterich, T. G. e Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286. Citado nas páginas 16 e 22.

Domingos, P. e Richardson, M. (2001). Mining the network value of customers. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 57–66, New York, NY, USA. ACM. Citado na página 5.

Drucker, H., Schapire, R. E., e Simard, P. (1993). Boosting performance in neural networks. *IJPRAI*, 7(4):705–719. Citado na página 19.

Fawcett, T., Foster, e Provost, F. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1:291–316. Citado na página 5.

Fawcett, T. e Provost, F. (1999). Activity monitoring: noticing interesting changes in behavior. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '99, páginas 53–62, New York, NY, USA. ACM. Citado na página 65.

Frank, A. e Asuncion, A. (2010). UCI machine learning repository. Citado nas páginas 36, 40, 42, 45, e 62.

Freund, Y. (1990). Boosting a weak learning algorithm by majority. In *COLT '90: Proceedings of the third annual workshop on Computational learning theory*, páginas 202–216, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. Citado na página 19.

Freund, Y. e Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT '95: Proceedings of the Second European Conference on Computational Learning Theory*, páginas 23–37, London, UK. Springer-Verlag. Citado nas páginas 13, 20, e 35.

Freund, Y. e Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, páginas 148–156. Citado nas páginas 13, 35, e 45.

Friedman, M. (1937). The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32(200):675–701. Citado nas páginas 61 e 69.

Friedman, M. (1940). A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *The Annals of Mathematical Statistics*, 11(1):86–92. Citado nas páginas 61 e 69.

Galstyan, A. e Cohen, P. (2005). Is guilt by association a bad thing. *Proceedings of the First International Conference on Intelligence Analysis*, 1(1). Citado na página 5.

García-Pedrajas, N. e Ortiz-Boyer, D. (2009). Boosting k-nearest neighbor classifier by means of input space projection. *Expert Syst. Appl.*, 36:10570–10582. Citado na página 44.

Girvan, M. e Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826. Citado na página 65.

Goadrich, M. e of Wisconsin Madison, T. U. (2007). *Learning ensembles of first-order clauses that optimize precision-recall curves*. The University of Wisconsin - Madison. Citado na página 5.

Hansen, L. e Salamon, P. (1990a). Neural network ensembles. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(10):993 –1001. Citado na página 10.

Hansen, L. K. e Salamon, P. (1990b). Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12:993–1001. Citado na página 5.

Hastie, T., Tibshirani, R., e Friedman, J. H. (2003). *The Elements of Statistical Learning*. Springer, corrected[a] edição. Citado na página 1.

Hosmer, D. W. e Lemeshow, S. (2000). *Applied logistic regression (Wiley Series in probability and statistics)*. Wiley-Interscience Publication. Citado nas páginas 29 e 32.

John, G. H. e Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Uncertainty in Artificial Intelligence*, páginas 338–345. Citado na página 61.

Kearns, M. e Valiant, L. (1994). Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, 41(1):67–95. Citado na página 19.

Kearns, M. e Valiant, L. G. (1988). Learning boolean formulae or finite automata is as hard as factoring. Relatório Técnico TR 14-88, Harvard University Aiken Computation Laboratory. Citado na página 19.

Leigh, W., Purvis, R., e Ragusa, J. M. (2002). Forecasting the nyse composite index with technical analysis, pattern recognizer, neural networks, and genetic algorithm: a case study in romantic decision support. *Decis. Support Syst.*, 32:361–377. Citado na página 2.

Lin, H.-J., Kao, Y.-T., Yang, F.-W., e Wang, P. S. P. (2006). Content-based image retrieval trained by adaboost for mobile application. Citado na página 2.

Lu, Q. e Getoor, L. (2003). Link-based classification. In Fawcett, T., Mishra, N., Fawcett, T., e Mishra, N., editors, *ICML*, páginas 496–503. AAAI Press. Citado nas páginas 27, 29, e 32.

Macskassy, S. A. e Provost, F. (2004). Classification in networked data: A toolkit and a univariate case study. Citado nas páginas 40 e 45.

Macskassy, S. A. e Provost, F. (2005). Suspicion scoring based on guilt-by-association, collective inference, and focused data access. In *International Conference on Intelligence Analysis*. Citado na página 5.

Macskassy, S. A. e Provost, F. (2007). Classification in networked data: A toolkit and a univariate case study. *J. Mach. Learn. Res.*, 8:935–983. Citado nas páginas 27, 30, 64, 65, e 87.

Maimon, O. e Rokac, L. (2004). Ensemble of decision trees for mining manufacturing data sets. Citado na página 2.

Mangiameli, P., West, D., e Rampal, R. (2004). Model selection for medical diagnosis decision support systems. *Decis. Support Syst.*, 36:247–259. Citado na página 2.

Margineantu, D. D. e Dietterich, T. G. (1997). Pruning adaptive boosting. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, páginas 211–218, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. Citado na página 77.

Meir, R. e Rätch, G. (2003). An introduction to boosting and leveraging. In *Advanced Lectures on Machine Learning*, LNCS, páginas 119–184. Springer. Citado na página 19.

Merkwirth, C., Mauser, H., Gasch, T. S., Roche, O., Stahl, M., e Lengauer, T. (2004). Ensemble methods for classification in cheminformatics. *J Chem Inf Comput Sci*, 44(6):1971–1978. Citado na página 2.

Michalski, R. S. (1983). A theory and methodology of inductive learning. *Artificial Intelligence*, 20:111–161. Citado na página 3.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York. Citado na página 1.

Motta, R. C. d. (2009). Uso de redes complexas na classificação relacional. Dissertação de Mestrado, ICMC - Universidade de São Paulo. Citado nas páginas xi e 4.

Murrugarra-Llerena, N. e de Andrade Lopes, A. (2011a). An adaptive graph-based k-nearest neighbor. In *Proceedings of the CoLISD: Collective Learning and Inference on Structured Data*, páginas 37–48, Atenas, Grecia. European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. Citado nas páginas 8 e 86.

Murrugarra-Llerena, N. e de Andrade Lopes, A. (2011b). A graph-based bagging. In *Proceedings of the CoLISD: Collective Learning and Inference on Structured Data*, páginas 25–36, Atenas, Grecia. European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. Citado nas páginas 7 e 86.

Neville, J. e Jensen, D. (2003). Collective classification with relational dependency networks. *Journal of Machine Learning Research*, 8:2007. Citado na página 5.

Newman, M. E. J. (2006). Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104. Citado na página 65.

Parmanto, B., Munro, P. W., e Doyle, H. R. (1995). Improving committee diagnosis with resampling techniques. In *NIPS*, páginas 882–888. Citado na página 14.

Paulovich, F. V., Oliveira, M. C. F., e Minghim, R. (2007). The projection explorer: A flexible tool for projection-based multidimensional visualization. In *Proceedings of the XX Brazilian Symposium on Computer Graphics and Image Processing - SIBGRAPI*, páginas 27–36, Belo Horizonte, Brazil. IEEE CS Press. Citado nas páginas 6, 8, 36, 42, e 87.

Platt, J. C. (1999). *Fast training of support vector machines using sequential minimal optimization*, páginas 185–208. MIT Press, Cambridge, MA, USA. Citado na página 61.

Plotkin, G. D. (1970). A Note on Inductive Generalization. *Machine Intelligence*, 5:153–163. Citado na página 3.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. Citado na página 61.

Ricci, F. e Aha, D. W. (1997). Extending local learners with error-correcting output codes. Relatório técnico, Naval Center for Applied Research in Artificial Inteligence, Washington. Citado na página 16.

Rokach, L. (2010). *Pattern classification using ensemble methods*. Series in machine perception and artificial intelligence. World Scientific. Citado nas páginas 12 e 13.

Schapire, R. E. (1989). The strength of weak learnability. *Machine Learning*, 5(2):197–227. Citado na página 19.

Schapire, R. E. (1997). Using output codes to boost multiclass learning problems. In *Proc. 14th International Conference on Machine Learning*, páginas 313–321. Morgan Kaufmann. Citado na página 13.

Schapire, R. E., Freund, Y., Bartlett, P., e Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686. Citado nas páginas 13 e 16.

Sun, S. e Huang, R. (2010). An adaptive k-nearest neighbor algorithm. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on*, volume 1, páginas 91–94. Citado na página 62.

Tan, A. C. e Gilbert, D. (2003). Multi-class protein fold classification using a new ensemble machine learning approach. *Genome Informatics*, 14:206–217. Citado na página 2.

Tang, E. K., Suganthan, P. N., e Yao, X. (2006). An analysis of diversity measures. *Mach. Learn.*, 65:247–271. Citado na página 18.

Taskar, B., Segal, E., e Koller, D. (2001). Probabilistic classification and clustering in relational data. In *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, páginas 870–878. Citado na página 5.

Tumer, K. e Ghosh, J. (1996). Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8(3/4):385–404. Citado na página 15.

Tumulty, K. (2006). Inside bush secret spynet [electronic version]. http://www.time.com/time/archive/preview/0,10987,1194021,00.html. Citado na página 5.

Valiant, L. G. (1984). A theory of the learnable. *Commun. ACM*, 27(11):1134–1142. Citado na página 19.

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83. Citado nas páginas 64 e 73.

Witten, I. H. e Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second[a] edição. Citado nas páginas 17, 20, e 25.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2):241–259. Citado na página 21.

Zhu, X. (2005). *Semi-Supervised Learning with Graphs*. Tese de Doutorado, Language Technologies Institute School of Computer Science - Carnegie Mellon University. Citado na página 4.

APÊNDICE

A

Tabelas com as comparações do A-KNN

Este apêndice contém as tabelas comparativas do método A-KNN. Em primeiro lugar, a Tabela A.1 apresenta os resultados comparativos do KNN com o A-KNN. Em segundo lugar, na Tabela A.2 são apresentados os resultados comparativos do A-KNN com o *Boosting* usando como classificador base o KNN. Finalmente, a Tabela A.3 apresenta resultados comparativos dos algoritmos Naive Bayes (NB), SVM, C4.5 (versões sem *Boosting* e com *Boosting*) e o A-KNN. Adicionalmente, é comparado o A-KNN com o *AdaNN* (uma abordagem de KNN adaptativo).

Tabela A.1: Resultados comparativos entre as acurácias do KNN (k = 1, 3, 5, 7, 9, 15, 20, 30 e 40) e A-KNN considerando 22 conjuntos de dados, cada um com dois níveis de ruído. Cada resultado é seguido pelo *ranking* da acurácia de cada algoritmo em cada conjunto de dados.

		K**NN**								
Conjunto de Dados	**A-KNN**	**k=1**	**k=3**	**k=5**	**k=7**	**k=9**	**k=15**	**k=20**	**k=30**	**k=40**
ba	89.07(7)	86.72(10)	86.74(9)	87.97(8)	89.71(4)	90.39(1)	90.14(2)	90.1(3)	89.28(5)	89.17(6)
ba (5%)	87.54(7)	86.07(10)	86.08(9)	87.11(8)	88.87(6)	89.91(1)	89.83(2)	89.71(3)	89.06(4)	89.04(5)
ba (10%)	85.94(8)	85.17(10)	85.19(9)	86.32(7)	88.42(6)	89.71(1)	89.67(2)	89.51(3)	88.99(5)	89.1(4)
bl	76.33(8)	71.06(10)	74.77(9)	77.18(6)	77.1(7)	77.31(5)	78.26(3)	78.92(1)	78.57(2)	78.09(4)
bl (5%)	75.99(8)	70.63(10)	74.57(9)	76.75(7)	76.99(6)	77.21(5)	77.92(3.5)	78.25(2)	78.45(1)	77.92(3.5)
bl (10%)	75.47(8)	70.63(10)	74.49(9)	76.24(7)	76.8(6)	76.91(5)	77.74(2)	77.61(3)	78.52(1)	77.34(4)
bt	69.57(2)	70.4(1)	66.85(3)	66.7(4)	65.53(5)	64.51(6)	61.4(7)	60.05(8)	54.2(9)	52.28(10)
bt (5%)	69.77(2)	70.4(1)	66.85(3)	66.7(4)	65.53(5)	64.51(6)	61.4(7)	60.05(8)	54.2(9)	52.28(10)
bt (10%)	66.91(2)	67.41(1)	66(4)	66.81(3)	65.04(5)	63.59(6)	61.99(7)	59.49(8)	52.27(9)	51.18(10)
ec	86.67(3)	80.66(9)	84.4(7)	86.1(4)	86.73(2)	86.9(1)	85.07(5)	84.41(6)	83.49(8)	80.42(10)
ec (5%)	85.87(4)	79.32(10)	84.02(7)	85.92(3)	86.25(2)	86.49(1)	85.19(5)	84.35(6)	83.73(8)	80.6(9)
ec (10%)	85.06(4)	77.65(10)	83.51(7)	85.47(3)	85.89(2)	86.1(1)	84.92(5)	84.39(6)	83.32(8)	80.43(9)
gl	68.42(3)	70.3(1)	69.84(2)	66.18(4)	63.05(5)	62.39(8)	62.86(6.5)	62.86(6.5)	61.28(9)	60.29(10)
gl (5%)	68.19(3)	69.28(2)	69.33(1)	66.46(4)	63.19(5)	63.18(6)	62.64(8)	62.96(7)	61.61(9)	59.86(10)
gl (10%)	66.48(3)	67.03(2)	68.17(1)	66.08(4)	63.42(5)	62.82(6)	62.4(8)	62.49(7)	61.14(9)	59.86(10)
hr	54.98(2)	69(1)	47.4(5)	34.48(9)	32.26(10)	40.01(8)	53.19(3)	49.43(4)	40.8(7)	41.25(6)
hr (5%)	54.27(2)	68.62(1)	47.25(5)	35.07(9)	33.18(10)	40.81(6)	53.9(3)	50.16(4)	39.81(7)	39.61(8)
hr (10%)	52.65(2)	65.8(1)	46.4(5)	34.14(9)	33.02(10)	40.32(7)	51.58(3)	48.05(4)	40.51(6)	38.59(8)
hs	80.93(5)	76.15(10)	79.11(9)	79.89(8)	80.81(6)	80.15(7)	81.37(4)	82.44(3)	83.07(2)	83.7(1)
hs (5%)	80.56(5)	74.33(10)	78(9)	79.56(8)	80.33(6)	80.04(7)	81.26(4)	82.33(3)	83.33(2)	83.37(1)
hs (10%)	79.41(7)	72.56(10)	76.78(9)	78.74(8)	80(6)	80.59(5)	80.74(4)	81.63(3)	83.07(2)	83.22(1)
io	83.42(8)	87.1(1)	86.02(2)	85.1(3)	84.3(4.5)	84.3(4.5)	84.25(6)	84.05(7)	79.6(9)	75.36(10)
io (5%)	82.6(8)	85.33(1)	85.13(2)	84.59(3)	84.02(4)	83.99(5)	83.68(6)	83.36(7)	79.12(9)	74.73(10)
io (10%)	81.63(8)	82.86(5)	84.28(1)	83.96(2)	83.77(3)	83.33(4)	82.77(6)	82.45(7)	78.58(9)	73.98(10)
ir	96.13(2.5)	95.4(8)	95.2(9)	95.73(4)	96.4(1)	95.47(7)	96.13(2.5)	95.6(5)	95.53(6)	94.27(10)
ir (5%)	95.73(4)	91.8(10)	94.8(8)	95.67(5)	96.13(1)	95.4(6)	96(2)	95.8(3)	95.2(7)	94.13(9)
ir (10%)	95.13(7)	86.27(10)	92.73(9)	95.2(6)	95.87(1.5)	95.4(4.5)	95.8(3)	95.87(1.5)	95.4(4.5)	94(8)
li	80.31(3)	86.06(1)	80.67(2)	75.78(4)	72.78(5)	68.03(6)	56.89(7)	54.86(8)	51.19(9)	48.89(10)
li (5%)	77.94(3)	82.67(1)	79.5(2)	75.39(4)	72.53(5)	67.14(6)	56.33(7)	53.61(8)	50.64(9)	47.83(10)
li (10%)	75.47(3)	79.14(1)	77.56(2)	74.83(4)	72.28(5)	67.56(6)	56.42(7)	53.92(8)	50.75(9)	47.39(10)
pd	74.08(5)	70.62(10)	73.86(6.5)	73.86(6.5)	74.45(2)	73.08(9)	74.38(3)	73.64(8)	74.48(1)	74.28(4)
pd (5%)	73.56(5)	69.55(10)	72.74(8)	73.02(7)	73.64(4)	72.45(9)	74.2(1)	73.28(6)	73.81(2)	73.77(3)
pd (10%)	72.61(5)	68.02(10)	71.26(9)	71.86(8)	72.56(6)	72.01(7)	73.36(2)	72.87(4)	73.55(1)	73.3(3)
se	96.15(2)	97.15(1)	96.12(3)	95.25(4)	94.89(5)	94.72(6)	94.36(7)	93.97(8)	93.1(9)	91.81(10)
se (5%)	95.53(1)	92.52(9)	95.39(2)	95.05(3)	94.8(4)	94.67(5)	94.31(6)	93.93(7)	93.04(8)	91.73(10)
se (10%)	94.4(2)	87.67(10)	93(7)	94.3(4)	94.35(3)	94.52(1)	94.08(5)	93.78(6)	92.94(8)	91.71(9)
so	84.64(2)	86.17(1)	83.76(3)	82.28(4)	79.1(5)	75.11(6)	68.77(10)	69.97(9)	71.27(7)	71.18(8)
so (5%)	83.09(3)	83.42(1)	83.41(2)	81.5(4)	78.43(5)	74.48(6)	69.19(10)	69.57(9)	70.74(7)	70.27(8)
so (10%)	81.5(2)	79.53(4)	82.02(1)	80.55(3)	77.38(5)	74.2(6)	69.88(7)	69.86(8)	69.67(9)	69.35(10)
st	53.48(4)	58.02(1)	57.01(3)	57.89(2)	51.82(6)	52.89(5)	51.56(7)	50.07(8)	41.7(9)	38.1(10)
st (5%)	54.91(4)	56.3(2)	56.03(3)	56.87(1)	52.22(6)	52.77(5)	50.32(7)	49.02(8)	42.47(9)	38.68(10)
st (10%)	53.3(4)	53.52(3)	54.69(2)	55.47(1)	52.37(5)	52.26(6)	49.07(7)	47.96(8)	42.19(9)	38.56(10)
te	56.67(2)	62.89(1)	48.23(9)	50.88(5)	51.73(3)	48.38(7)	51.54(4)	49.89(6)	46.46(10)	48.37(8)
te (5%)	55.66(2)	61.63(1)	49.36(7)	50.41(6)	50.79(4)	48.66(8)	51.55(3)	50.75(5)	47.78(10)	48.58(9)
te (10%)	54.47(2)	59.42(1)	49.09(6)	48.7(7)	49.3(5)	48.66(8)	50.96(3)	49.5(4)	47.74(9)	47.67(10)
ve	74.53(2)	75.71(1)	72.71(4)	73.69(3)	72.51(5)	71.35(6)	70.15(8)	70.23(7)	68.78(9)	66.41(10)
ve (5%)	73.07(2)	72.4(3)	71.5(5)	73.11(1)	72.21(4)	70.75(6)	69.64(8)	69.86(7)	68.02(9)	66.24(10)
ve (10%)	71.63(2)	68.91(8)	69.73(5)	72.09(1)	71.41(3)	69.99(4)	69.15(6)	69.02(7)	67.35(9)	65.72(10)
vo	95.69(3)	98.86(1)	96.89(2)	94.23(4)	90.24(5)	85.41(6)	72.87(7)	68.56(8)	64.98(9)	62.24(10)
vo (5%)	93.1(4)	94.25(2)	95.79(1)	93.68(3)	89.23(5)	84.15(6)	72.59(7)	68.44(8)	64.69(9)	61.6(10)
vo (10%)	89.6(3)	88.87(4)	93.21(1)	92.11(2)	88.15(5)	83.07(6)	72.08(7)	68.38(8)	63.96(9)	60.41(10)
wi	97.59(2)	95.12(10)	95.85(7)	95.5(9)	95.9(6)	95.83(8)	97.08(3)	97.7(1)	96.85(4)	96.62(5)
wi (5%)	96.91(2)	91.93(10)	95.23(9)	95.39(8)	95.61(7)	95.72(6)	96.74(5)	97.35(1)	96.86(3)	96.8(4)
wi (10%)	96.4(5)	87.18(10)	92.98(9)	94.5(8)	95.01(7)	95.51(6)	96.69(3.5)	97.25(1)	96.8(2)	96.69(3.5)
wqr	63.21(2)	64.38(1)	57.75(10)	58.35(5)	57.76(8.5)	57.76(8.5)	58.16(6)	59.11(3)	58.99(4)	58.09(7)
wqr (5%)	62.96(2)	63.16(1)	57.2(10)	58.04(6)	57.5(9)	57.66(8)	58.02(7)	58.78(4)	58.82(3)	58.09(5)
wqr (10%)	62.73(1)	61.97(2)	56.49(10)	57.44(8)	57.14(9)	57.69(7)	57.98(6)	58.7(4)	58.77(3)	58.14(5)
wbc	96.31(7)	95.28(10)	96.6(4)	96.93(1)	96.67(2)	96.65(3)	96.58(5)	96.4(6)	96.2(8)	95.97(9)
wbc (5%)	96(8)	93.81(10)	96.1(7)	96.82(1)	96.64(3)	96.74(2)	96.51(4)	96.41(5)	96.15(6)	95.95(9)
wbc (10%)	95.39(8)	92.05(10)	95.15(9)	96.28(5)	96.42(3)	96.57(1)	96.5(2)	96.38(4)	96.15(6)	95.95(7)
ye	59.15(1)	52.61(10)	55.09(9)	57.07(8)	58.18(7)	58.66(4)	58.31(6)	59(2)	58.77(3)	58.42(2)
ye (5%)	57.7(7)	51.62(10)	54.51(9)	56.87(8)	58.1(6)	58.46(3)	58.15(5)	58.92(1)	58.52(2)	58.26(4)
ye (10%)	51.76(9)	50.36(10)	53.59(8)	56.27(7)	57.61(6)	58(3.5)	57.92(5)	58.73(1)	58.22(2)	58(3.5)
zo	95.54(2)	96.05(1)	92.61(4)	95.05(3)	90.68(5)	88.9(6)	88.53(7)	83.18(8)	74.79(9)	68.24(10)
zo (5%)	95.24(2)	96.05(1)	92.61(4)	95.05(3)	90.68(5)	88.9(6)	88.53(7)	83.18(8)	74.79(9)	68.24(10)
zo (10%)	87.65(7)	92.41(2.5)	92.41(2.5)	94.07(1)	89.9(4)	88.91(5)	88.42(6)	83.48(8)	74.7(9)	67.06(10)
Média dos *Ranks*	4.053	5.3106	5.5909	4.8864	5.0227	5.3485	5.197	5.4394	6.4924	7.6591
Ranks	1	5	8	2	3	6	4	7	9	10

Tabela A.2: Resultados comparativos das acurácias da versão *Boosting* do KNN (k = 1, 3, 5, 7, 9, 15, 20, 30 e 40) e o algoritmo A-KNN considerando 22 conjuntos de dados, cada um com dois níveis de ruído. Cada resultado é seguido pelo *ranking* da acurácia de cada algoritmo em cada conjunto de dados.

		K**NN**								
Conjunto de dados	**A-KNN**	**k=1**	**k=3**	**k=5**	**k=7**	**k=9**	**k=15**	**k=20**	**k=30**	**k=40**
ba	89.07(7)	86.72(10)	86.74(9)	87.97(8)	89.71(4)	90.39(1)	90.06(3)	90.1(2)	89.35(5)	89.17(6)
ba (5 %)	87.54(7)	86.07(10)	86.08(9)	87.11(8)	88.87(6)	89.91(1)	89.83(2)	89.71(3)	89.06(4)	89.04(5)
ba (10 %)	85.94(8)	85.17(10)	85.19(9)	86.32(7)	88.42(6)	89.71(1)	89.67(2)	89.51(3)	88.99(5)	89.1(4)
bl	76.33(8)	70.7(10)	73.6(9)	77.11(6)	77.06(7)	77.31(5)	78.26(3)	78.92(1)	78.57(2)	78.09(4)
bl (5 %)	75.99(8)	70.46(10)	73.46(9)	76.59(7)	76.99(6)	77.21(5)	77.92(3.5)	78.25(2)	78.45(1)	77.92(3.5)
bl (10 %)	75.47(8)	70.31(10)	73.45(9)	76.23(7)	76.8(6)	76.91(5)	77.74(2)	77.61(3)	78.52(1)	77.34(4)
bt	69.57(3)	70.4(2)	65.88(4)	70.46(1)	65.02(5)	64.51(6)	61.4(7)	60.05(8)	54.38(9)	51.9(10)
bt (5 %)	69.77(3)	70.4(2)	65.88(4)	70.46(1)	65.02(5)	64.51(6)	61.4(7)	60.05(8)	54.38(9)	51.9(10)
bt (10 %)	66.91(3)	67.41(2)	64.36(5)	67.58(1)	65.24(4)	63.59(6)	61.99(7)	59.49(8)	52.37(9)	51.27(10)
ec	86.67(3)	80.66(10)	84.14(7)	85.57(4)	86.73(2)	86.9(1)	85.07(5)	84.41(6)	83.49(8)	83.23(9)
ec (5 %)	85.87(3)	79.32(10)	83.69(8)	85.48(4)	86.25(2)	86.49(1)	85.19(5)	84.35(6)	83.73(7)	82.54(9)
ec (10 %)	85.06(4)	77.65(10)	82.26(8)	85.39(3)	85.89(2)	86.1(1)	84.92(5)	84.39(6)	83.32(7)	80.54(9)
gl	68.42(2)	70.3(1)	68.22(3)	65.47(4)	62.72(7)	62.57(8)	63.7(5)	63.55(6)	61.18(9)	60.29(10)
gl (5 %)	68.19(2)	69.23(1)	66.6(3)	65.66(4)	63.19(6)	63.18(7)	63.14(8)	63.42(5)	61.65(9)	59.86(10)
gl (10 %)	66.48(2)	66.93(1)	65.24(4)	65.44(3)	63.52(5)	62.82(7)	62.54(8)	62.91(6)	61.14(9)	59.86(10)
hr	54.98(3)	69(1)	52.03(4)	36.75(9)	32.57(10)	40.68(8)	58.32(2)	51.18(5)	40.8(7)	41.25(6)
hr (5 %)	54.27(3)	68.62(1)	51.57(5)	36.65(9)	34.25(10)	41.17(6)	58.05(2)	51.91(4)	39.81(7)	39.69(8)
hr (10 %)	52.65(3)	65.8(1)	49.98(4)	35.2(9)	34.07(10)	40.62(6)	53.92(2)	49(5)	40.51(7)	38.59(8)
hs	80.93(4)	76.15(10)	77.3(7)	76.78(8)	76.19(9)	78.26(6)	80.7(5)	82.44(3)	83.07(2)	83.7(1)
hs (5 %)	80.56(5)	74.33(10)	75.04(9)	76(8)	76.67(7)	79.07(6)	81.15(4)	82.3(3)	83.33(2)	83.37(1)
hs (10 %)	79.41(6)	72.56(10)	73.15(9)	75.26(8)	77.74(7)	79.7(5)	80.7(4)	81.48(3)	83.07(2)	83.22(1)
io	83.42(10)	87.1(5)	87.52(3)	87.41(4)	86.66(6)	86.27(7)	89.69(1)	88.89(2)	86.24(8)	85.38(9)
io (5 %)	82.6(10)	85.08(3)	83.87(6)	83.53(8)	82.9(9)	83.57(7)	85.76(1.5)	85.76(1.5)	84.02(4)	83.88(5)
io (10 %)	81.63(6)	82.31(5)	81.03(8)	80.72(10)	80.77(9)	81.62(7)	84.19(1)	83.85(2)	83.62(4)	83.68(3)
ir	96.13(1)	95.4(5.5)	95.13(9)	95.67(3.5)	95.67(3.5)	95.4(5.5)	96(2)	95.2(7.5)	95.2(7.5)	93.13(10)
ir (5 %)	95.73(3)	91.73(10)	93.93(9)	95.07(6)	94.73(7)	95.2(4)	96(1)	95.87(2)	95.13(5)	94.13(8)
ir (10 %)	95.13(4)	86.07(10)	88.2(9)	92.47(8)	93.87(7)	94.47(5)	95.73(1.5)	95.73(1.5)	95.4(3)	94.07(6)
li	80.31(3)	86.06(1)	83.53(2)	77.83(4)	72.83(5)	71.08(6)	61.25(7)	57.14(8)	52.11(9)	49.56(10)
li (5 %)	77.94(3)	82.69(1)	78.86(2)	75.33(4)	72.58(5)	69.19(6)	59.78(7)	55.03(8)	51.03(9)	48.33(10)
li (10 %)	75.47(2)	79.17(1)	75.44(3)	72.47(4)	72.44(5)	67.86(6)	58.56(7)	54.36(8)	50.92(9)	47.33(10)
pd	74.08(5)	70.62(10)	73.8(7)	73.86(6)	74.45(2)	73.08(9)	74.38(3)	73.64(8)	74.48(1)	74.28(4)
pd (5 %)	73.56(5)	69.55(10)	72.68(8)	73.02(7)	73.64(4)	72.45(9)	74.2(1)	73.28(6)	73.81(2)	73.77(3)
pd (10 %)	72.61(5)	68.02(10)	71.26(9)	71.86(8)	72.56(6)	72.01(7)	73.36(2)	72.87(4)	73.54(1)	73.16(3)
se	96.15(2)	97.15(1)	96.13(3)	95.03(4)	94.55(6)	94.65(5)	94.34(7)	93.77(8)	93.08(9)	91.9(10)
se (5 %)	95.53(1)	92.45(7)	90.27(10)	90.88(9)	93.47(5)	94.67(2)	94.31(3)	93.93(4)	93.04(6)	91.73(8)
se (10 %)	94.4(2)	87.57(9)	84.82(10)	88.48(8)	94.18(3)	94.52(1)	94.08(4)	93.78(5)	92.94(6)	91.71(7)
so	84.64(3)	86.17(1)	84.72(2)	82.62(4)	81.27(5)	80.06(6)	69.95(10)	70.35(9)	73.27(7)	71.95(8)
so (5 %)	83.09(2)	83.42(1)	82.36(3)	80.88(4)	79.09(5)	77.98(6)	69.82(9)	69.51(10)	71.12(7)	69.92(8)
so (10 %)	81.5(1)	79.53(2)	79.33(3)	78.81(4)	77.71(5)	75.07(6)	69.92(8)	70.3(7)	69.15(9.5)	69.15(9.5)
st	53.48(4)	58.02(2)	57.23(3)	61.07(1)	53.44(5)	53.26(6)	51.14(7)	49.88(8)	41.7(9)	38.1(10)
st (5 %)	54.91(4)	56.3(3)	56.37(2)	58.26(1)	52.68(6)	53.24(5)	50.01(7)	49.03(8)	42.36(9)	38.68(10)
st (10 %)	53.3(4)	53.52(3)	55.13(2)	56.86(1)	52.69(5)	52.16(6)	48.74(7)	47.52(8)	42.19(9)	38.56(10)
te	56.67(2)	63.88(1)	51.77(3)	51.14(6)	51.73(4)	48.38(9)	51.67(5)	50.23(7)	46.86(10)	48.63(8)
te (5 %)	55.66(2)	62.48(1)	52.73(3)	50.34(7)	50.86(6)	48.26(9)	51.68(4)	50.88(5)	47.79(10)	48.71(8)
te (10 %)	54.47(2)	60.49(1)	51.01(3)	49.03(7)	49.36(6)	48.39(8)	50.96(4)	49.83(5)	47.8(9)	47.74(10)
ve	74.53(2)	75.71(1)	71.88(5)	73.69(3)	72.51(4)	71.35(6)	70.15(8)	70.23(7)	68.78(9)	66.41(10)
ve (5 %)	73.07(2)	72.43(3)	68.14(8)	73.11(1)	72.21(4)	70.75(5)	69.64(7)	69.86(6)	68.02(9)	66.24(10)
ve (10 %)	71.63(2)	69.07(6)	65.1(10)	72.09(1)	71.41(3)	69.99(4)	69.15(5)	69.02(7)	67.35(8)	65.72(9)
vo	95.69(3)	98.86(1)	97.08(2)	95.59(4)	92.42(5)	88.4(6)	77.44(7)	71(8)	65.05(9)	62.29(10)
vo (5 %)	93.1(3)	94.25(2)	94.35(1)	91.71(4)	88.72(5)	84.27(6)	73.48(7)	68.94(8)	64.69(9)	61.76(10)
vo (10 %)	89.6(2)	88.87(3)	91.03(1)	88.62(4)	87.7(5)	83.07(6)	72.31(7)	68.44(8)	63.96(9)	60.6(10)
wi	97.59(1)	95.12(8)	95.85(6)	94.43(10)	95.11(9)	95.21(7)	96.67(4)	97.47(2)	96.34(5)	97.08(3)
wi (5 %)	96.91(1)	91.93(7)	90.96(9)	89.78(10)	91.47(8)	93.3(6)	96.22(5)	96.73(3)	96.68(4)	96.79(2)
wi (10 %)	96.4(4)	87.18(9)	87.49(8)	85.88(10)	90.83(7)	92.02(6)	96.13(5)	97.08(1)	96.8(2)	96.69(3)
wqr	63.21(2)	64.38(1)	56.15(10)	58.35(5)	57.76(8.5)	57.76(8.5)	58.16(6)	59.11(3)	58.99(4)	58.09(7)
wqr (5 %)	62.96(2)	63.13(1)	55.16(10)	58.04(6)	57.5(9)	57.66(8)	58.02(7)	58.78(4)	58.82(3)	58.09(5)
wqr (10 %)	62.73(1)	61.88(2)	54.29(10)	57.44(8)	57.14(9)	57.69(7)	57.98(6)	58.7(4)	58.77(3)	58.14(5)
wbc	96.31(4.5)	95.28(6)	96.58(3)	96.93(1)	96.67(2)	96.31(4.5)	93.76(8)	93.33(9)	93.05(10)	93.94(7)
wbc (5 %)	96(7)	93.81(10)	95.58(9)	96.42(2)	96.64(1)	96.24(5)	96.38(4)	96.4(3)	96.15(6)	95.95(8)
wbc (10 %)	95.39(8)	92.05(10)	94.12(9)	95.74(7)	95.98(5)	96.28(3)	96.41(1)	96.38(2)	96.15(4)	95.95(6)
ye	59.15(1)	52.61(10)	53(9)	57.07(8)	58.18(7)	58.66(4)	58.31(6)	59(2)	58.77(3)	58.42(5)
ye (5 %)	57.7(7)	51.62(10)	52.51(9)	56.87(8)	58.1(6)	58.46(3)	58.15(5)	58.92(1)	58.52(2)	58.26(4)
ye (10 %)	51.76(9)	50.34(10)	52.52(8)	56.27(7)	57.61(6)	58(3.5)	57.92(5)	58.73(1)	58.22(2)	58(3.5)
zo	95.54(2)	96.05(1)	91.73(6)	94.16(4)	92.18(5)	95.01(3)	87.14(8)	88.12(7)	80.14(9)	77.93(10)
zo (5 %)	95.24(2)	96.05(1)	91.73(6)	94.16(4)	92.08(5)	95.11(3)	87.05(8)	88.32(7)	80.14(9)	78.03(10)
zo (10 %)	87.65(7)	91.33(2)	89.72(5)	92.69(1)	90.31(3)	90.02(4)	88.42(6)	83.78(8)	75.99(9)	73.88(10)
Média dos *Ranks*	3.8409	5.1439	6.1212	5.3864	5.6364	5.3636	4.947	5.1439	6.2424	7.1742
Ranks	1	3.5	8	6	7	5	2	3.5	9	10

Tabela A.3: Resultados comparativos das acurácias dos algoritmos C4.5, SVM, Naive Bayes (NB) (versão sem *Boosting* e com *Boosting*) e A-KNN considerando 22 conjuntos de dados, cada um com dois níveis de ruído. Cada resultado é seguido pelo *ranking* da acurácia de cada algoritmo em cada conjunto de dados. Além disso, a comparação com uma abordagem de KNN adaptativo ($adaNN$) é apresentada.

	Sem *Boosting*				**Com *Boosting***					
Conjunto de Dados	**A-KNN**	**NB**	**C4.5**	**SVM**	**A-KNN**	**NB**	**C4.5**	**SVM**	**A-KNN**	**AdaNN**
ba	89.07(2)	90.53(1)	77.82(4)	87.57(3)	89.07(2)	91.68(1)	78.35(4)	87.54(3)	89.07	88.47
ba (5%)	87.54(2)	89.91(1)	78.69(4)	87.38(3)	87.54(2)	89.91(1)	77.82(4)	87.38(3)	87.54	88.39
ba (10 %)	85.94(3)	89.7(1)	79.17(2)	87.24(4)	85.94(3)	89.7(1)	77.18(4)	87.24(2)	85.94	87.92
bl	76.33(2)	75.28(4)	78.2(1)	76.18(3)	76.33(4)	77.01(2)	77.43(1)	76.99(3)	76.33	76.56
bl (5%)	75.99(3)	75.26(4)	77.9(1)	76.17(2)	75.99(4)	76.81(2)	77.27(1)	76.68(3)	75.99	76.27
bl (10 %)	75.47(3.5)	75.47(3.5)	77.7(1)	76.21(2)	75.47(4)	76.43(3)	77.33(1)	76.7(2)	75.47	75.78
bt	69.57(1)	67.75(2)	65.41(3)	60.08(4)	69.57(1)	67.75(3)	68.91(2)	60.48(4)	69.57	69.5
bt (5%)	69.77(1)	67.75(2)	65.41(3)	59.44(4)	69.77(1)	67.75(3)	68.91(2)	60.11(4)	69.77	69.5
bt (10 %)	66.91(1)	61.3(3)	64.6(2)	58.01(4)	66.91(1)	61.3(3)	63.77(2)	58.32(4)	66.91	66.1
ec	86.67(1)	85.5(2)	82.83(4)	83.48(3)	86.67(1)	85.5(2)	82.62(4)	84.69(3)	86.67	86.76
ec (5%)	85.87(1)	85.44(2)	83.28(3)	82.83(4)	85.87(1)	85.44(2)	82.14(4)	83.06(3)	85.87	86.25
ec (10 %)	85.06(2)	85.08(1)	81.94(4)	82.32(3)	85.06(2)	85.08(1)	81.01(4)	82.32(3)	85.06	85.38
gl	68.42(1)	47.75(4)	67.61(2)	57.72(3)	68.42(2)	47.75(4)	75.83(1)	59.53(3)	68.42	68.43
gl (5%)	68.19(1)	44.59(4)	67.79(2)	57.49(3)	68.19(2)	44.59(4)	75.66(1)	58.32(3)	68.19	68.34
gl (10 %)	66.48(2)	44.04(4)	68.35(1)	57.62(3)	66.48(2)	44.04(4)	73.02(1)	58.34(3)	66.48	67.4
hr	54.98(4)	73.63(2)	79.44(1)	55.54(3)	54.98(4)	76.07(2)	84.73(1)	58.15(3)	54.98	53.05
hr (5%)	54.27(4)	71.08(2)	81.45(1)	54.48(3)	54.27(4)	74.42(2)	83.12(1)	57.74(3)	54.27	52.59
hr (10 %)	52.65(4)	67.54(2)	81.2(1)	54.65(3)	52.65(4)	71.48(2)	79.75(1)	56.8(3)	52.65	50.96
hs	80.93(3)	83.59(2)	78.15(4)	83.89(1)	80.93(3)	82.59(2)	78.59(4)	83.85(1)	80.93	79.33
hs (5%)	80.56(3)	84(1)	77.81(4)	83.48(2)	80.56(3)	82.7(2)	76.59(4)	83.19(1)	80.56	78.48
hs (10 %)	79.41(3)	84.37(1)	76.11(4)	83.44(2)	79.41(3)	83.04(2)	75.85(4)	83.3(1)	79.41	78
io	83.42(3)	82.17(4)	89.74(1)	88.07(2)	83.42(4)	91.06(2)	93.05(1)	89.29(3)	83.42	84.62
io (5%)	82.6(3)	82.53(4)	88.38(1)	87.89(2)	82.6(4)	91.29(1)	90.8(2)	88.32(3)	82.6	83.79
io (10 %)	81.63(4)	82.22(3)	87.27(1)	86.89(2)	81.63(4)	90.55(1)	87.92(2)	87.03(3)	81.63	82.4
ir	96.13(2)	95.53(3)	94.73(4)	96.27(1)	96.13(2)	94.8(3)	94.33(4)	97.73(1)	96.13	96.27
ir (5%)	95.73(2)	94.93(3)	94.07(4)	96.07(1)	95.73(2)	95(3)	91.8(4)	96.07(1)	95.73	95.87
ir (10 %)	95.13(2)	93.4(3)	92.8(4)	95.87(1)	95.13(2)	93.4(3)	87.87(4)	95.87(1)	95.13	94.73
li	80.31(1)	64.14(4)	69.36(3)	73.83(2)	80.31(1)	64.5(4)	78.97(2)	77.92(3)	80.31	80.53
li (5%)	77.94(1)	62.75(4)	66.19(3)	73.67(2)	77.94(1)	62.75(4)	76.28(2)	74.03(3)	77.94	78.31
li (10 %)	75.47(1)	61.94(4)	63.78(3)	72.56(2)	75.47(1)	61.94(4)	73.28(2)	72.5(3)	75.47	75.69
pd	74.08(4)	75.75(2)	74.49(3)	76.8(1)	74.08(3)	75.86(2)	71.69(4)	76.8(1)	74.08	75.13
pd (5%)	73.56(3)	75.56(2)	73.41(4)	76.97(1)	73.56(3)	75.77(2)	72.22(4)	76.97(1)	73.56	74.52
pd (10 %)	72.61(4)	75.01(2)	73.44(3)	76.85(1)	72.61(3)	75.06(2)	71.09(4)	76.85(1)	72.61	73.17
se	96.15(2)	80.11(4)	96.74(1)	92.91(3)	96.15(2)	80.11(4)	98.16(1)	92.91(3)	96.15	96.42
se (5%)	95.53(2)	75.6(4)	96.1(1)	92.22(3)	95.53(1)	75.7(4)	95.52(2)	92.22(3)	95.53	95.82
se (10 %)	94.4(2)	72.82(4)	94.77(1)	91.44(3)	94.4(1)	72.89(4)	92.49(2)	91.44(3)	94.4	94.49
so	84.64(1)	67.71(4)	73.61(3)	76.6(2)	84.64(1)	80.77(2)	79.13(3)	78.22(4)	84.64	84.25
so (5%)	83.09(1)	68.1(4)	72.01(3)	76.55(2)	83.09(1)	78.87(2)	77.5(3)	74.67(4)	83.09	82.85
so (10 %)	81.5(1)	68.2(4)	70.6(3)	74.99(2)	81.5(1)	76.23(2)	75.1(3)	73.84(4)	81.5	81.02
st	53.48(2)	42.61(4)	62.04(1)	52.19(3)	53.48(3)	43.78(4)	66.17(1)	55.88(2)	53.48	52.96
st (5%)	54.91(2)	43.28(4)	62.42(1)	49.94(3)	54.91(2)	44.26(4)	64.88(1)	53.64(3)	54.91	52.68
st (10 %)	53.3(2)	42.22(4)	61.91(1)	50.31(3)	53.3(2)	42.77(4)	62.58(1)	52.53(3)	53.3	52.23
te	56.67(2)	53.41(4)	57.41(1)	53.6(3)	56.67(2)	53.35(4)	65.52(1)	53.6(3)	56.67	55.35
te (5%)	55.66(1)	53.61(3)	55.05(2)	52.33(4)	55.66(2)	53.61(3)	61.32(1)	52.6(4)	55.66	53.88
te (10 %)	54.47(1)	52.08(3)	53.26(2)	51.84(4)	54.47(2)	52.22(3)	54.66(1)	51.71(4)	54.47	52.74
ve	74.53(3)	45.4(4)	76.18(1)	74.57(2)	74.53(3)	45.4(4)	81.09(1)	74.59	74.53	74.23
ve (5%)	73.07(3)	45.83(4)	74.41(1)	73.75(2)	73.07(3)	45.83(4)	78.81(1)	73.75(2)	73.07	73.1
ve (10 %)	71.63(3)	45.14(4)	71.83(2)	72.97(1)	71.63(3)	45.14(4)	75.72(1)	72.97(2)	71.63	71.82
vo	95.69(1)	66.82(4)	79.82(3)	68.84(2)	95.69(1)	74.23(3)	92.47(2)	69.24(4)	95.69	97.55
vo (5%)	93.1(1)	64.86(4)	78.16(2)	66.54(3)	93.1(1)	66.91(3)	89.84(2)	66.54(4)	93.1	94.83
vo (10 %)	89.6(1)	62.79(4)	74.94(2)	63.73(3)	89.6(1)	63.32(4)	85.68(2)	63.73(3)	89.6	91.17
wi	97.59(2)	97.46(3)	93.2(4)	98.76(1)	97.59(2)	96.18(4)	96.45(3)	97.98(1)	97.59	95.07
wi (5%)	96.91(2)	96.9(3)	90.8(4)	98.18(1)	96.91(1)	94.08(4)	94.57(3)	96.05(2)	96.91	95.07
wi (10 %)	96.4(2)	96.1(3)	87.64(4)	97.4(1)	96.4(2)	95.88(3)	91.73(4)	96.84(1)	96.4	94.56
wqr	63.21(1)	54.84(4)	60.78(2)	58.21(3)	63.21(2)	54.84(4)	67.38(1)	58.21(3)	63.21	62.06
wqr (5%)	62.96(1)	54.8(4)	60.34(2)	57.97(3)	62.96(2)	54.8(4)	66.2(1)	57.97(3)	62.96	61.85
wqr (10 %)	62.73(1)	54.8(4)	59.67(2)	57.8(3)	62.73(2)	54.8(4)	65.19(1)	57.8(3)	62.73	61.54
wbc	96.31(2)	96.07(3)	95.01(4)	96.75(1)	96.31(2)	95.55(4)	96.08(3)	96.72(1)	96.31	96.21
wbc (5%)	96(3)	96.27(2)	94.42(4)	96.54(1)	96(3)	96.29(2)	94.26(4)	96.54(1)	96	95.97
wbc (10 %)	95.39(3)	96.39(1)	94.16(4)	96.28(2)	95.39(3)	96.4(1)	93.05(4)	96.28(2)	95.39	95.64
ye	59.15(1)	57.99(2)	56.61(4)	57.03(3)	59.15(1)	57.99(2)	56.75(4)	57.03(3)	59.15	58.92
ye (5%)	57.7(2)	58.01(1)	55.56(4)	56.29(3)	57.7(2)	58.01(1)	56.1(4)	56.29(3)	57.7	58.75
ye (10 %)	51.76(4)	57.64(1)	54.37(3)	55.33(2)	51.76(4)	57.64(1)	55.05(3)	55.33(2)	51.76	58.43
zo	95.54(2)	96.95(1)	92.61(4)	93.68(3)	95.54(4)	96.95(1)	96.25(3)	96.82(2)	95.54	93.66
zo (5%)	95.24(2)	96.95(1)	92.61(4)	93.48(3)	95.24(4)	96.95(1)	96.25(3)	96.52(2)	95.24	93.57
zo (10 %)	87.65(4)	94.58(1)	92.11(3)	93.29(2)	87.65(4)	90.49(3)	92.68(1)	92.67(2)	87.65	86.67
Média dos *Ranks*	2.1288	2.8864	2.5758	2.4091	2.3182	2.7273	2.3788	2.5758	-	-
Ranks	1	4	3	2	1	4	2	3	-	-

APÊNDICE

B

Tabelas com comparações em Dados Relacionais

Este apêndice contém as tabelas comparativas de classificadores baseados em grafos e métodos *ensembles*. Em primeiro lugar, as Tabelas B.1 e B.2 apresentam os resultados comparativos de classificadores baseados em grafos e sua versão *Bagging* baseado em Grafos com Duplicação de Arestas (BGDA). Em segundo lugar, as Tabelas B.3 e B.4 apresentam os resultados comparativos de classificadores baseados em grafos e sua versão *Bagging* baseado em Grafos com Remoção de Vértices Duplicados (BGRVD). Seguidamente, as Tabelas B.5 e B.6 apresentam os resultados comparativos de classificadores baseados em grafos e sua versão *Bagging* baseado em Grafos com Pesos (BGP). Finalmente, as Tabelas B.7 e B.8 apresentam os resultados comparativos de classificadores baseados em grafos e sua versão *Cross-Validated Committees* (CVCG).

Tabela B.1: Resultados comparativos das acurácias dos algoritmos wvrn, prn, cdrn e nbc; e sua versão *Bagging* com Duplicação de Arestas (BGDAwvrn, BGDAprn, BGDAcdrn e BGDAnbc). Cada resultado é seguido do teste de Wilcoxon.

	wvrn	BGDAwvrn	prn	BGDAprn	cdrn	BGDAcdrn	nbc	BGDAnbc
adj	20.38	**23.86**	26.59	**33.86**	**80.53**	77.80	82.20	**84.02**
foot	**92.12**	90.38	**91.29**	89.47	**86.82**	**86.82**	**86.82**	85.98
iall	73.63	**73.77**	70.79	**71.00**	77.59	**78.07**	**61.90**	60.79
iprod	76.06	**77.72**	75.30	**76.20**	**82.17**	81.06	**81.61**	81.33
ipr	**54.27**	53.09	51.58	**52.35**	**54.73**	53.50	**40.70**	39.74
iyh	64.18	**64.79**	59.12	**60.73**	**64.46**	64.13	**48.67**	47.05
pblogs	**92.35**	**92.35**	91.74	**93.02**	**94.23**	94.03	93.89	**93.96**
pbooks	**88.73**	85.91	**85.82**	84.00	**85.82**	**85.82**	**86.73**	85.82
wcc	**61.25**	58.98	38.75	**49.33**	58.43	**58.44**	**41.88**	41.31
wcl	23.92	**31.34**	29.06	**30.49**	39.65	**49.02**	43.90	**45.88**
wtc	**73.73**	72.83	46.83	**68.07**	**72.24**	69.55	55.31	**55.61**
wtl	14.49	**17.48**	18.63	**18.66**	44.71	**49.21**	**50.64**	49.16
wwac	66.15	**66.84**	49.31	**66.82**	69.34	**69.36**	**57.39**	55.08
wwal	40.11	**41.95**	43.31	**44.48**	49.09	**54.12**	**64.73**	64.05
wwic	**74.59**	73.74	50.85	**66.68**	70.91	**72.90**	63.57	**64.43**
wwil	**22.04**	22.03	22.04	**22.32**	49.66	**49.94**	**54.79**	53.10
Wilcoxon	-	-	$\triangledown$	$\triangle$	-	-	-	-

Tabela B.2: Resultados comparativos das acurácias dos algoritmos nlb-m, nlb-d, nlb-c e nlb-b; e sua versão *Bagging* com Duplicação de Arestas (BGDAnlb-m, BGDAnlb-d, BGDAnlb-c e BGDAnlb-b). Cada resultado é seguido do teste de Wilcoxon.

	nlb-m	BGDAnlb-m	nlb-d	BGDAnlb-d	nlb-c	BGDAnlb-c	nlb-b	BGDAnlb-b
adj	81.44	**82.35**	**85.98**	85.91	**88.71**	87.65	63.41	**65.23**
foot	**91.21**	87.65	**89.55**	87.80	**87.80**	**87.80**	25.45	**38.48**
iall	74.53	**78.00**	**79.53**	78.98	**84.60**	83.83	**56.77**	**56.77**
iprod	77.87	**80.01**	**82.38**	82.10	79.53	**79.60**	62.53	**63.98**
ipr	**53.82**	53.13	**53.08**	51.94	45.59	**45.46**	41.62	**42.62**
iyh	**64.85**	63.24	**62.74**	61.79	58.68	**60.12**	38.43	**40.99**
pblogs	92.21	**92.48**	**94.23**	94.09	**94.23**	94.09	60.67	**62.75**
pbooks	**86.82**	84.91	**86.73**	85.73	84.73	**85.73**	62.82	**62.91**
wcc	62.68	**63.26**	**66.68**	**66.68**	**67.24**	66.13	**46.15**	**46.15**
wcl	41.88	**45.87**	54.72	**57.27**	54.13	**55.27**	43.91	**48.47**
wtc	**72.82**	55.61	78.42	**79.01**	79.32	**81.38**	45.59	**46.77**
wtl	**50.02**	49.16	**53.29**	51.83	60.42	**60.71**	45.00	**47.99**
wwac	**68.22**	55.08	**71.21**	70.52	73.95	**74.17**	**44.69**	44.46
wwal	61.28	**64.05**	**69.80**	68.42	**68.18**	67.72	**52.33**	51.86
wwic	**73.47**	64.43	79.11	**79.65**	77.99	**81.37**	**47.75**	**47.75**
wwil	52.53	**53.10**	**55.66**	55.65	**60.42**	56.48	48.86	**49.42**
Wilcoxon	-	-	-	-	-	-	$\triangledown$	$\triangle$

Tabela B.3: Resultados comparativos das acurácias dos algoritmos wvrn, prn, cdrn e nbc; e sua versão *Bagging* com Remoção de Vértices Duplicados (BGRVDwvrn, BGRVDprn, BGRVDcdrn e BGRVDnbc). Cada resultado é seguido do teste de Wilcoxon.

	wvrn	BGRVDwvrn	prn	BGRVDprn	cdrn	BGRVDcdrn	nbc	BGRVDnbc
adj	20.38	**23.86**	26.59	**33.86**	**80.53**	77.80	82.20	**84.02**
foot	**92.12**	90.38	**91.29**	89.47	**86.82**	**86.82**	**86.82**	85.98
iall	**73.63**	73.15	70.79	**71.34**	77.59	**78.07**	**61.90**	60.79
iprod	76.06	**77.38**	75.30	**75.78**	**82.17**	81.06	**81.61**	81.33
ipr	**54.27**	53.09	51.58	**52.35**	**54.73**	53.50	**40.70**	39.74
iyh	64.18	**64.79**	59.12	**60.73**	**64.46**	64.13	**48.67**	47.05
pblogs	92.35	**92.68**	91.74	**92.75**	**94.23**	94.03	93.89	**93.96**
pbooks	**88.73**	85.91	**85.82**	84.00	**85.82**	**85.82**	**86.73**	85.82
wcc	**61.25**	58.98	38.75	**49.33**	58.43	**58.44**	**41.88**	41.31
wcl	23.92	**31.34**	29.06	**30.49**	39.65	**49.02**	43.90	**45.88**
wtc	**73.73**	72.83	46.83	**68.07**	**72.24**	69.55	55.31	**55.61**
wtl	14.49	**17.18**	18.63	**19.55**	44.71	**49.21**	**50.64**	49.16
wwac	66.15	**66.84**	49.31	**66.82**	69.34	**69.36**	**57.39**	55.08
wwal	40.11	**41.95**	43.31	**44.48**	49.09	**54.12**	**64.73**	64.05
wwic	**74.59**	73.46	50.85	**66.68**	70.91	**72.90**	63.57	**64.43**
wwil	22.04	**21.75**	**22.04**	**22.04**	49.66	**49.94**	**54.79**	53.10
Wilcoxon	-	-	$\triangledown$	$\triangle$	-	-	-	-

Tabela B.4: Resultados comparativos das acurácias dos algoritmos nlb-m, nlb-d, nlb-c e nlb-b; e sua versão *Bagging* com Remoção de Vértices Duplicados (BGRVDnlb-m, BGRVDnlb-d, BGRVDnlb-c e BGRVDnlb-b). Cada resultado é seguido do teste de Wilcoxon.

	nlb-m	BGRVDnlb-m	nlb-d	BGRVDnlb-d	nlb-c	BGRVDnlb-c	nlb-b	BGRVDnlb-b
adj	**81.44**	80.53	**85.98**	85.91	**88.71**	87.65	63.41	**65.23**
foot	**91.21**	87.65	**89.55**	87.80	**87.80**	**87.80**	25.45	**38.48**
iall	74.53	**77.93**	**79.53**	78.98	**84.60**	83.83	**56.77**	**56.77**
iprod	77.87	**80.15**	**82.38**	82.10	79.53	**79.60**	62.53	**63.98**
ipr	**53.82**	52.72	**53.08**	51.94	45.59	**45.46**	41.62	**42.62**
iyh	**64.85**	63.63	**62.74**	61.79	58.68	**60.12**	38.43	**40.99**
pblogs	92.21	**92.62**	**94.23**	94.09	**94.23**	94.09	60.67	**62.75**
pbooks	**86.82**	84.91	**86.73**	85.73	84.73	**85.73**	62.82	**62.91**
wcc	**62.68**	62.13	**66.68**	**66.68**	**67.24**	66.13	**46.15**	**46.15**
wcl	41.88	**46.73**	54.72	**57.27**	54.13	**55.27**	43.91	**48.47**
wtc	**72.82**	72.22	78.42	**79.01**	79.32	**81.38**	45.59	**46.77**
wtl	50.02	**51.24**	**53.29**	51.83	60.42	**60.71**	45.00	**47.99**
wwac	68.22	**68.66**	**71.21**	70.52	73.95	**74.17**	**44.69**	44.46
wwal	61.28	**61.97**	**69.80**	68.42	**68.18**	67.72	**52.33**	51.86
wwic	73.47	**75.15**	79.11	**79.65**	77.99	**81.37**	**47.75**	**47.75**
wwil	**52.53**	51.72	**55.66**	55.65	**60.42**	56.48	48.86	**49.42**
Wilcoxon	-	-	-	-	-	-	$\triangledown$	$\triangle$

Tabela B.5: Resultados comparativos das acurácias dos algoritmos wvrn, prn, cdrn e nbc; e sua versão *Bagging* com Pesos (BGPwvrn, BGPprn, BGPcdrn e BGPnbc). Cada resultado é seguido do teste de Wilcoxon.

	wvrn	BGPwvrn	prn	BGPprn	cdrn	BGPcdrn	nbc	BGPnbc
adj	20.38	**22.20**	26.59	**27.58**	**80.53**	76.97	82.20	**83.11**
foot	**92.12**	**92.12**	**91.29**	89.47	**86.82**	85.98	86.82	**86.89**
iall	**73.63**	**73.63**	70.79	**71.34**	77.59	**77.86**	**61.90**	61.56
iprod	76.06	**77.38**	75.30	**76.13**	**82.17**	81.33	**81.61**	81.33
ipr	**54.27**	53.09	51.58	**52.58**	**54.73**	53.41	**40.70**	40.66
iyh	64.18	**64.46**	59.12	**60.01**	**64.46**	63.79	48.67	**48.72**
pblogs	92.35	**93.42**	91.74	**92.82**	**94.23**	93.96	**93.89**	93.96
pbooks	**88.73**	87.73	85.82	**86.82**	85.82	**86.82**	**86.73**	**86.73**
wcc	**61.25**	59.83	38.75	**44.75**	**58.43**	58.15	**41.88**	41.60
wcl	23.92	**29.07**	29.06	**31.06**	39.65	**49.33**	**43.90**	43.32
wtc	**73.73**	73.13	46.83	**64.86**	**72.24**	70.74	55.31	**56.19**
wtl	14.49	**16.89**	**18.63**	17.16	44.71	**48.92**	**50.64**	50.36
wwac	66.15	**66.61**	49.31	**65.45**	69.34	**70.06**	**57.39**	56.92
wwal	40.11	**40.79**	**43.31**	42.40	49.09	**53.18**	**64.73**	63.14
wwic	**74.59**	72.61	50.85	**64.13**	70.91	**72.06**	63.57	**63.58**
wwil	**22.04**	20.90	**22.04**	21.47	49.66	**49.94**	**54.79**	52.83
Wilcoxon	-	-	$\bigtriangledown$	$\bigtriangleup$	-	-	-	-

Tabela B.6: Resultados comparativos das acurácias dos algoritmos nlb-m, nlb-d, nlb-c e nlb-b; e sua versão *Bagging* com Pesos (BGPnlb-m, BGPnlb-d, BGPnlb-c e BGPnlb-b). Cada resultado é seguido do teste de Wilcoxon.

	nlb-m	BGPnlb-m	nlb-d	BGPnlb-d	nlb-c	BGPnlb-c	nlb-b	BGPnlb-b
adj	81.44	**82.27**	**85.98**	85.83	**88.71**	87.65	63.41	**65.23**
foot	**91.21**	88.56	**89.55**	88.64	87.80	**90.45**	25.45	**38.48**
iall	74.53	**78.63**	**79.53**	79.25	**84.60**	82.03	**56.77**	**56.77**
iprod	77.87	**80.50**	**82.38**	81.75	**79.53**	78.21	62.53	**63.98**
ipr	**53.82**	52.63	**53.08**	51.85	**45.59**	41.89	41.62	**42.62**
iyh	**64.85**	63.35	**62.74**	62.12	**58.68**	58.24	38.43	**40.99**
pblogs	92.21	**92.62**	**94.23**	93.96	**94.23**	93.69	60.67	**62.75**
pbooks	**86.82**	85.82	**86.73**	85.82	84.73	**84.82**	62.82	**62.91**
wcc	62.68	**62.96**	**66.68**	66.40	**67.24**	65.56	**46.15**	**46.15**
wcl	41.88	**47.29**	54.72	**56.71**	**54.13**	53.00	43.91	**48.47**
wtc	**72.82**	71.92	78.42	**79.61**	79.32	**80.49**	45.59	**46.77**
wtl	50.02	**50.62**	**53.29**	51.83	**60.42**	56.84	45.00	**47.99**
wwac	68.22	**69.81**	**71.21**	**71.20**	**73.95**	73.72	**44.69**	44.46
wwal	61.28	**63.58**	**69.80**	68.43	**68.18**	66.58	**52.33**	51.86
wwic	73.47	**73.73**	**79.11**	78.82	77.99	**80.25**	**47.75**	**47.75**
wwil	**52.53**	50.87	55.66	**55.92**	**60.42**	55.94	48.86	**49.42**
Wilcoxon	-	-	-	-	-	-	$\bigtriangledown$	$\bigtriangleup$

Tabela B.7: Resultados comparativos das acurácias dos algoritmos wvrn, prn, cdrn e nbc; e sua versão *cross-validated committees* (CVCGwvrn, CVCGprn, CVCGcdrn e CVCGnbc). Cada resultado é seguido do teste de Wilcoxon.

	wvrn	CVCGwvrn	prn	CVCGprn	cdrn	CVCGcdrn	nbc	CVCGnbc
adj	**20.38**	17.65	26.59	**28.48**	80.53	**82.42**	**82.20**	**82.20**
foot	**92.12**	91.21	**91.29**	89.47	86.82	**87.65**	**86.82**	**86.82**
iall	**73.63**	73.35	70.79	**71.34**	77.59	**77.86**	**61.90**	61.84
iprod	76.06	**77.45**	75.30	**78.00**	**82.17**	**82.17**	**81.61**	81.33
ipr	54.27	**54.59**	51.58	**52.72**	**54.73**	54.55	**40.70**	40.48
iyh	64.18	**64.91**	59.12	**60.12**	64.46	**64.69**	**48.67**	48.39
pblogs	92.35	**93.42**	91.74	**93.42**	94.23	**94.30**	93.89	**94.03**
pbooks	**88.73**	**88.73**	85.82	**86.82**	**85.82**	**85.82**	**86.73**	**86.73**
wcc	61.25	**61.53**	38.75	**45.03**	**58.43**	**58.43**	**41.88**	41.60
wcl	23.92	**28.21**	29.06	**29.35**	39.65	**42.80**	43.90	**44.47**
wtc	**73.73**	73.43	46.83	**61.90**	**72.24**	71.93	55.31	**55.32**
wtl	**14.49**	12.43	**18.63**	15.67	44.71	**46.19**	50.64	**51.52**
wwac	66.15	**66.84**	49.31	**66.14**	**69.34**	69.12	**57.39**	**57.39**
wwal	40.11	**40.80**	43.31	**43.77**	**49.09**	49.08	64.73	**65.18**
wwic	**74.59**	74.03	50.85	**64.42**	70.91	**71.77**	**63.57**	63.28
wwil	**22.04**	19.52	**22.04**	20.63	49.66	**50.23**	54.79	**55.07**
Wilcoxon	-	-	$\triangledown$	$\triangle$	$\triangledown$	$\triangle$	-	-

Tabela B.8: Resultados comparativos das acurácias dos algoritmos nlb-m, nlb-d, nlb-c e nlb-b; e sua versão *Cross-Validated Committees* (CVCGnlb-m, CVCGnlb-d, CVCGnlb-c e CVCGnlb-b). Cada resultado é seguido do teste de Wilcoxon.

	nlb-m	CVCGnlb-m	nlb-d	CVCGnlb-d	nlb-c	CVCGnlb-c	nlb-b	CVCGnlb-b
adj	81.44	**82.42**	85.98	**86.82**	**88.71**	87.80	**63.41**	**63.41**
foot	**91.21**	89.47	**89.55**	**89.55**	87.80	**89.47**	25.45	**29.02**
iall	74.53	**76.75**	79.53	**79.60**	**84.60**	84.53	**56.77**	**56.77**
iprod	77.87	**80.08**	**82.38**	82.31	79.53	**79.60**	**62.53**	**62.53**
ipr	53.82	**53.86**	53.08	**53.31**	**45.59**	45.46	41.62	**41.98**
iyh	**64.85**	64.63	**62.74**	62.68	58.68	**59.68**	38.43	**39.60**
pblogs	92.21	**92.95**	**94.23**	**94.23**	**94.23**	**94.23**	60.67	**62.42**
pbooks	**86.82**	85.82	86.73	**87.73**	**84.73**	**84.73**	**62.82**	55.27
wcc	62.68	**63.54**	**66.68**	66.12	**67.24**	66.95	46.15	**46.44**
wcl	41.88	**43.31**	54.72	**56.15**	54.13	**54.98**	**43.91**	43.63
wtc	72.82	**74.00**	78.42	**78.72**	79.32	**80.51**	**45.59**	**45.59**
wtl	50.02	**50.94**	**53.29**	52.70	60.42	**60.43**	45.00	**46.18**
wwac	68.22	**68.44**	71.21	**71.44**	73.95	**75.11**	**44.69**	**44.69**
wwal	61.28	**62.90**	**69.80**	69.34	**68.18**	67.50	**52.33**	51.87
wwic	73.47	**74.03**	79.11	**79.67**	77.99	**78.56**	**47.75**	**47.75**
wwil	**52.53**	50.56	55.66	**56.22**	**60.42**	59.85	**48.86**	**48.86**
Wilcoxon	-	-	-	-	-	-	-	-