# Novas aborgagens em aprendizado de máquina para a geração de regras, classes desbalanceadas e ordenação de casos

***Ronaldo Cristiano Prati***

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura:_____

# Novas aborgagens em aprendizado de máquina para a geração de regras, classes desbalanceadas e ordenação de casos[1]

***Ronaldo Cristiano Prati***

**Orientadora:** *Profª Drª Maria Carolina Monard*

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação — ICMC/USP — como parte dos requisitos necessários à obtenção do título de Doutor em Ciências de Computação e Matemática Computacional.

**USP - São Carlos**
**Julho/2006**

[1]Trabalho realizado com auxílio financeiro da CAPES.

à João, Maria, Maria Carolina e Silvia

# Agradecimentos

Primeiramente gostaria de agradecer aos meus pais por, em suas simplicidades, me mostrarem que respeito, dignidade e honestidade são os principais valore que um homem deve ter em sua vida. Eles sempre me apoiaram e incentivaram em todas as minhas decisões, mesmo tendo que fazer vários sacrifícios para que eu pudesse realizar meus sonhos. Também gostaria de agradecer aos meus irmãos Lucas e Evandro e a minha avó Palmira.

Gostaria de agradecer à professora Maria Carolina, minha orientadora, pelo carinho, amizade, confiança e orientação, tanto acadêmica quanto na vida. A senhora pode ter certeza que o carinho, amizade e admiração são sentimentos mútuos de minha parte, e que é para mim um exemplo de vida.

Também gostaria de agradecer à minha noiva Silvia. Cada dia que passa eu tenho mais certeza que você é a companheira ideal, sempre leal, carinhosa e amiga. Obrigado por me incentivar sempre, mesmo quando nós ficamos separados pela distância, mas sempre unidos em pensamento. Estendo meus agradecimentos aos seus pais, Silvio e Alda, por me acolherem dentro de sua casa.

Gostaria de agradecer aos meus amigos Gustavo, Edson, Huei e Richardson, e para os outros inúmeros amigos do LABIC, pela amizade e companherismo desses anos todos. Agradeço também aos outros moradores da república aqui em São Carlos, Mauro, Sidney, Sayuri, Márcio e Daniel pelos bons momentos.

Também gostaria de agradecer ao Professor Peter Flach e demais membros do seu grupo de pesquisa pelo ano que passei lá na Universidade de Bristol. Esse ano foi muito importante para o meu amadurecimento profissional e pessoal.

Finalmente gostaria de agradecer a CAPES pelo apoio financeiro, tanto no Brasil quanto no ano em que estive fora; a pró-reitoria de pesquisa pelo apoio na participação de congressos; e a Universidade de São Paulo, em especial ao Instituto de Ciências Matemáticas e de Computação, pelo suporte e estrutura disponibilizados para o desenvolvimento desta tese.

# Resumo

Algoritmos de aprendizado de máquina são frequentemente os mais indicados em uma grande variedade de aplicações de mineração dados. Entretanto, a maioria das pesquisas em aprendizado de máquina refere-se ao problema bem definido de encontrar um modelo (geralmente de classificação) de um conjunto de dados pequeno, relativamente bem preparado para o aprendizado, no formato atributo-valor, no qual os atributos foram previamente selecionados para facilitar o aprendizado. Além disso, o objetivo a ser alcançado é simples e bem definido (modelos de classificação precisos, no caso de problemas de classificação). Mineração de dados propicia novas direções para pesquisas em aprendizado de máquina e impõe novas necessidades para outras. Com a mineração de dados, algoritmos de aprendizado estão quebrando as restrições descritas anteriormente. Dessa maneira, a grande contribuição da área de aprendizado de máquina para a mineração de dados é retribuída pelo efeito inovador que a mineração de dados provoca em aprendizado de máquina. Nesta tese, exploramos alguns desses problemas que surgiram (ou reaparecem) com o uso de algoritmos de aprendizado de máquina para mineração de dados. Mais especificamente, nos concentramos seguintes problemas:

**Novas abordagens para a geração de regras.** Dentro dessa categoria, propomos dois novos métodos para o aprendizado de regras. No primeiro, propomos um novo método para gerar regras de exceção a partir de regras gerais. No segundo, propomos um algoritmo para a seleção de regras denominado ROCCER. Esse algoritmo é baseado na análise ROC. Regras provêm de um grande conjunto externo de regras e o algoritmo proposto seleciona regras baseado na região convexa do gráfico ROC.

**Proporção de exemplos entre as classes.** Investigamos vários aspectos relacionados a esse tópico. Primeiramente, realizamos uma série de experimentos em conjuntos de dados artificiais com o objetivo de testar nossa hipótese de que o grau de sobreposição entre as classes é um fator complicante em conjuntos de dados muito desbalanceados. Também executamos uma extensa análise experimental com vários métodos (alguns deles propostos neste trabalho) para balancear artificialmente conjuntos de dados desbalanceados. Finalmente, investigamos o relacionamento entre classes desbalanceadas e pequenos disjuntos, e a influência da proporção de classes no processo de rotulação de exemplos no algoritmo de aprendizado de máquina semi-supervisionado Co-training.

**Novo método para a combinação de rankings.** Propomos um novo método, chamado BORDARANK, para construir *ensembles* de *rankings* baseado no método de votação borda count. BORDARANK pode ser aplicado em qualquer problema de ordenação binária no qual vários rankings estejam disponíveis. Resultados experimentais mostram uma melhora no desempenho com relação aos *rankings* individuais, alem de um desempenho comparável com algoritmos mais sofisticados que utilizam a predição numérica, e não *rankings*, para a criação de *ensembles* para o problema de ordenação binária.

# Abstract

Machine learning algorithms are often the most appropriate algorithms for a great variety of data mining applications. However, most machine learning research to date has mainly dealt with the well-circumscribed problem of finding a model (generally a classifier) given a single, small and relatively clean dataset in the attribute-value form, where the attributes have previously been chosen to facilitate learning. Furthermore, the end-goal is simple and well-defined, such as accurate classifiers in the classification problem. Data mining opens up new directions for machine learning research, and lends new urgency to others. With data mining, machine learning is now removing each one of these constraints. Therefore, machine learning's many valuable contributions to data mining are reciprocated by the latter's invigorating effect on it. In this thesis, we explore this interaction by proposing new solutions to some problems due to the application of machine learning algorithms to data mining applications. More specifically, we contribute to the following problems.

**New approaches to rule learning.** In this category, we propose two new methods for rule learning. In the first one, we propose a new method for finding exceptions to general rules. The second one is a rule selection algorithm based on the ROC graph. Rules come from an external larger set of rules and the algorithm performs a selection step based on the current convex hull in the ROC graph.

**Proportion of examples among classes.** We investigated several aspects related to this issue. Firstly, we carried out a series of experiments on artificial data sets in order to verify our hypothesis that overlapping among classes is a complicating factor in highly skewed data sets. We also carried out a broadly experimental analysis with several methods (some of them proposed by us) that artificially balance skewed datasets. Our experiments show that, in general, over-sampling methods perform better than under-sampling methods. Finally, we investigated the relationship between class imbalance and small disjuncts, as well as the influence of the proportion of examples among classes in the process of labelling unlabelled cases in the semi-supervised learning algorithm Co-training.

**New method for combining rankings.** We propose a new method called BORDARANK to construct ensembles of rankings based on borda count voting, which could be applied whenever only the rankings are available. Results show an improvement upon the base-rankings constructed by taking into account the ordering given by classifiers which output continuous-valued scores, as well as a comparable performance with the fusion of such scores.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Algoritmos

# Lista de Exemplos

Capítulo 1

# Introdução

ESTA tese de doutoramento tem como tema aprendizado de máquina e uma de suas mais diretas aplicações, a descoberta de conhecimento a partir de dados. Neste capítulo é apresentada uma descrição da tese, com o intuito de fornecer ao leitor uma visão geral dos assuntos abordados neste trabalho, bem como algumas das soluções propostas. Este capítulo está organizado da seguinte maneira: na Seção 1.1 é apresentada a motivação deste trabalho, destacando os principais desafios para a utilização de aprendizado de máquina para a descoberta de conhecimento. Na Seção 1.2 são apresentados os objetivos desta tese de doutoramento. Na Seção 1.3 são destacadas as principais contribuições, as quais serão abordadas mais profundamente nos capítulos seguintes. Finalmente, na Seção 1.4 é apresentada a organização do trabalho, com uma breve descrição do conteúdo de cada capítulo.

## 1.1. Motivação

É freqüente encontrar definições (ao menos informais) de que Mineração de Dados (MD) é uma aplicação de Aprendizado de Máquina (AM). Uma das possíveis razões para essa confusão de conceitos é que muitas das fer-

ramentas utilizadas em MD estão baseadas em algoritmos de AM. Neste sentido é inegável que AM forme o núcleo do que se convencionou denominar por MD. Em contrapartida, devido a esse massivo uso de algoritmos de AM, mineração de dados é hoje uma das grandes vitrines para aprendizado de máquina. Por esse simples motivo, é inegável a existência de uma forte e sinergética sobreposição entre as duas áreas.

Entretanto, apesar das duas áreas terem uma grande semelhança e até mesmo compartilharem um sem número de métodos, práticas e objetivos, elas são áreas distintas, cada qual com seus avanços, limitações e desafios a serem superados. Essa sobreposição não pode ser vista simplesmente como uma relação de dependência de uma para com a outra. Ao menos do ponto de vista de áreas de pesquisa, o elemento comum a ser explorado reside no fato de que mineração de dados representa uma genuína fonte para alimentar as pesquisas em aprendizado de máquina, e vice-versa.

### 1.1.1 Sobre aprendizado de máquina

Em linhas gerais, aprendizado de máquina pode ser caracterizado por uma série de práticas voltadas para a solução de problemas para os quais geralmente não se conhece *a priori* uma solução ou modelagem capaz de resolvê-los. O que se conhece é um conjunto finito de fatos (também conhecidos como casos ou exemplos) que descrevem objetos, processos, situações, ou ambientes e o objetivo é encontrar alguma solução a partir desses fatos. Essas práticas incluem, entre outros, generalização a partir de um conjunto de casos cujas classes são conhecidas (Michie et al., 1994), reconhecimento de padrões (Duda et al., 2000), identificação de agrupamentos e correlações (Everitt et al., 2001), refinamento de teorias a partir de métodos de aprendizado (Džeroski & Lavrač, 2001) e aprendizado por realimentação dos estados do ambiente (Sutton & Barto, 1998).

Aprendizado de máquina tem uma grande sobreposição com estatística, uma vez que ambas as áreas estudam a análise de dados. Entretanto, diferentemente da estatística, que foca-se principalmente em modelos teóricos bem definidos e ajustamento de parâmetros a esses modelos, aprendizado de máquina tem um foco mais algorítmico, utilizando representações de modelos mais flexíveis e heurísticas para a realização de busca. Por exemplo, uma análise estatística pode determinar distribuições, covariâncias e correlações entre os atributos que descrevem os fatos, mas não é capaz de caracterizar essas dependências em um nível abstrato e conceitual como os humanos fazem, nem prover uma explicação causal do porque essas dependências existem. Enquanto uma análise estatística dos dados pode determinar as tendências centrais e variâncias de determinados fatores, ela

não pode produzir uma descrição qualitativa das regularidades, nem tão pouco determinar as dependências em fatores não providos explicitamente com os dados (Kaufman & Michalski, 2005).

A flexibilidade dos métodos de aprendizado de máquina os torna apropriados para situações nas quais existe pouco conhecimento *a priori* sobre o domínio, e/ou esse conhecimento *a priori* é de difícil elicitação, dificultando a escolha de modelos estatísticos. Essa flexibilidade também significa que algoritmos de AM são freqüentemente capazes de aprender com dados que não foram coletados a partir de um rigoroso processo experimental controlado, mas obtidos a partir de um processo qualquer cujo principal objetivo não seja a descoberta de conhecimento. O preço a pagar por essa flexibilidade é a falta de um poder analítico-teórico que a utilização dos modelos estatísticos pré-definidos possuem.

Outro ponto importante a ser ressaltado é que não existe um método de aprendizado de "propósito-geral". Cada método tem a sua utilidade comprometida pelas suas suposições e particularidades, e cada aplicação requer atenção especial para a definição dos métodos a serem utilizados. Além disso, análises teóricas de métodos de aprendizado são muito difíceis, e dificilmente se têm garantias de que os resultados estão corretos. Em contrapartida, aprendizado de máquina faz uso intensivo do poder computacional para validar experimentalmente seus métodos e resultados, utilizando métodos estatísticos de reamostragem, tal como a validação cruzada, além da validação experimental em conjuntos de dados de *benchmark* (Newman et al., 1998) e experimentação com conjuntos de dados especialmente designados para testar hipóteses específicas (Langley, 2000).

Representações de modelos construídos por algoritmos de aprendizado incluem várias formas de representação equivalentes à lógica proposicional e relacional, agrupamentos, hierarquias de conceitos, redes probabilísticas, entre outras. Métodos de busca geralmente utilizados incluem busca em feixe, gradiente descendente, maximização das expectativas, algoritmos genéticos e busca em profundidade, entre outros.

### 1.1.2 Sobre mineração de dados

Para se entender melhor o que é mineração de dados, é importante diferenciar MD do uso tradicional de computadores para o tratamento e análise de dados. Atualmente, dados são armazenados e organizados em sistemas gerenciadores de Bases de Dados (BD). Consultas e relatórios são ferramentas simples e práticas que podem ser utilizadas para explorar esses dados em vários níveis. Ferramentas de consultas a BDs recuperam a informação enquanto que ferramentas de relatórios apresentam essa informação

de uma maneira clara e organizada. Mineração de dados, em contrapartida, é utilizada para extrair informações interessantes ocultas nos dados. A principal diferença é que, ao contrário da mineração de dados, o critério para a realização da consulta a uma BD é estabelecido previamente. Mineração de dados, ao contrário, procura por relações e associações entre eventos que não são necessariamente conhecidas *a priori*. Para entender melhor a diferença, considere o exemplo a seguir.

O departamento de *marketing* de uma empresa quer saber se existe algum relacionamento entre dois produtos A e B. Para tanto, eles consultam as bases de dados de vendas dos produtos A e B separadamente e em conjunto por meio de uma consulta tal como: "Quantas vezes o produto A com o produto B foram comprados em conjunto?"; "Quantas vezes o produto A foi comprado?"; "Quantas vezes o produto B foi comprado?". Com essas informações, é possível determinar se há um relacionamento entre esses dois produtos A e B, mas somente entre eles. Essa consulta baseia-se na suposição de que existe alguma conexão entre a compra conjunta de A e B. Essa suposição deve ser formulada antes que a consulta seja feita. Compare com um exemplo típico de MD. A pergunta agora é: "Quais são os produtos que apresentam alguma conexão?", ou seja, quais produtos são mais freqüentemente comprados juntos. Nesse caso, serão encontradas as conexões mais freqüentes entre todos os pares de produtos na BD, já que nenhuma suposição foi feita *a priori* com relação aos produtos a serem consultados. Nesse caso, observe que a conexão entre os produtos A e B pode ou não ser encontrada, dependendo da freqüência com que esses produtos são adquiridos em conjunto. É claro que essas funcionalidades podem ser implementadas dentro de um gerenciador de bases de dados utilizando uma composição de consultas. Entretanto, essa não é uma consulta trivial e nem a atividade principal para a qual esses sistemas foram projetados.

Por prover os conjuntos de dados reais, mineração de dados é uma fonte natural para a aplicação de algoritmos de aprendizado de máquina. Muitos dos métodos de aprendizado de máquina mais poderosos têm suas aplicações mais visíveis em bases de dados reais disponíveis hoje em dia, as quais não estavam disponíveis quando os métodos foram originalmente propostos. Entretanto, a maioria das pesquisas em aprendizado de máquina relatadas na literatura está delimitada a problemas bem definidos, como criar um modelo de classificação dado um conjunto de dados simples, pequeno, relativamente bem preparado em uma tabela no formato atributo-valor, no qual os atributos foram previamente selecionados para facilitar o aprendizado e a meta é simples e bem definida, geralmente expressa como taxa de acerto de classificação.

Em mineração de dados, no entanto, aprendizado é raramente um processo isolado, mas uma série de atividades, que vão desde a definição do problema, preparação dos dados, aprendizado, até a utilização dos resultados obtidos. Nesse sentido, mineração de dados permitiu que aprendizado de máquina estendesse suas idéias e motivações em novas direções (Domingos, 2002). Com o advento da mineração de dados, pesquisas em aprendizado de máquina obtiveram um incentivo para quebrar muitos dos paradigmas, restrições e limitações existentes até então. Dessa maneira, as valiosas contribuições do aprendizado de máquina para a mineração de dados têm a reciprocidade pelo inerente caráter revigorante da mineração de dados. Nesta tese, procuramos explorar o benefício mútuo da interação entre essas duas áreas.

### 1.1.3 Problemas em aberto de AM relevantes para MD

Muitos métodos de AM assumem que os exemplos (casos) não contêm erros, que todos os valores dos atributos são conhecidos, que todos os exemplos provém de uma mesma base de dados e que existe um conceito preciso a ser aprendido que não muda com o tempo. Em muitas situações, muitas dessas restrições não são aplicáveis, uma vez que os dados disponíveis são dados reais (Kaufman & Michalski, 2005). Esse fato leva a uma série de problemas não contemplados em algoritmos de aprendizado de máquina tradicionais, e à necessidade do desenvolvimento de uma grande variedade de métodos para resolvê-los. Esses problemas incluem:

**Aprendizagem a partir de exemplos com ruído** para tratar de exemplos que contenham uma certa quantidade de erros ou ruído (Blum et al., 2003). Esses problemas são de grande importância em AM uma vez que dados reais freqüentemente contêm quantidades consideráveis de ruído;

**Aprendizagem a partir de dados incompletos** no qual deve-se levar em consideração exemplos para os quais os valores para alguns dos atributos são desconhecidos (Batista & Monard, 2003a);

**Aprendizagem com atributos irrelevante/reduntantes** no qual deve-se levar em consideração atributos que ou são irrelevantes, i.e., não contribuem na definição do conceito ou redundantes, i.e., existes atributos similares com um poder de descrição parecido Lee (2005); Liu & Motoda (1998).

**Aprendizagem com mudanças de conceito** no qual o conceito meta a ser aprendido não é estático, mas pode mudar com o tempo (Widmer &

Kubat, 1996). Essa mudança se dá ou aleatoriamente ou em alguma direção específica. Um exemplo típico de mudança de conceito é a previsão do tempo, no qual o conceito pode mudar radicalmente de acordo com a estação do ano.

**Aprendizagem incremental** na qual os exemplos chegam em um fluxo contínuo de dados e o modelo que os caracteriza pode necessitar de ajustes conforme chegam os dados.

**Aprendizagem com dados distribuídos temporal e espacialmente** a maioria dos algoritmos de aprendizado não é capaz de lidar com dados que são distribuídos temporal ou espacialmente. Nesses casos, os dados devem ser considerados em conjunto e as relações entre eles devem ser exploradas.

**Aprendizagem com dados viciados** a maioria dos dados reais não foram coletados com o intuito de se utilizar métodos de aprendizado. Dessa maneira, a suposição usual de dados independentes e identicamente distribuídos não é necessariamente verdade. Uma importante direção de pesquisas esta relacionada ao fato que a amostra não é independente e que dados do passado, presente e futuro não são necessariamente da mesma população (Ryabko, 2006).

**Aprendizagem com custos** a maioria dos algoritmos de aprendizado assume que todos os erros têm um mesmo custo, mas isso raramente acontece na prática (Elkan, 2001a). Um problema relacionado é o de classes desbalanceadas, uma vez que é fácil obter altas taxas de acerto favorecendo as classes mais freqüentes.

**Aprendizagem de padrões isolados** a meta tradicional de criar um modelo global do processo que representa os dados, muitas vezes deve ser trocada pela procura por padrões localizados ou desvios com relação a um padrão pré-estabelecido.

## 1.2. Objetivos

Como visto na seção anterior, aprendizado de máquina contribui com mineração de dados com a flexibilidade e expressividade na representação de modelos, a importância das validações empíricas e com o grande poder algoritmico-computacional baseado nos mais diferentes métodos e heurísticas de busca. Em contrapartida, mineração de dados requer que algoritmos de aprendizado sejam capazes de lidar com várias questões práticas, tais

como a escalabilidade a grandes bases de dados, utilização de custos no processo de aprendizagem, ambientes dinâmicos e conceitos que possam evoluir com o tempo, capacidade de lidar com classes de exemplos pouco freqüentes e novas formas de representação e organização de conhecimento. Em suma, mineração de dados abriu várias novas direções para pesquisas em aprendizado de máquina. Além disso, em vários aspectos, ela propicia uma visão renovada em problemas que receberam algum tratamento na área, mas que tiveram uma diminuição de interesse devido principalmente à falta de dados reais relevantes.

O principal objetivo desta tese é explorar alguns desses problemas que surgiram (ou reaparecem) com o uso de algoritmos de aprendizado de máquina para mineração de dados. Mais especificamente, durante o desenvolvimento desta tese, nos concentramos nos seguintes problemas:

1. Podemos explorar alguma representação alternativa para a indução de modelos? Como incorporar essa representação em um algoritmo de aprendizado? Preferencialmente, representações que sejam mais facilmente entendíveis por seres humanos (especialistas ou não).

2. Podemos explorar algum método de busca alternativo na geração de conjuntos de regras de conhecimento? Como o desempenho desse método se compara com outros métodos?

3. Conjuntos de dados com classes desbalanceadas podem piorar o desempenho de algoritmos de aprendizado? Como? Em que condições? Podemos compensar o problema causado pelas classes desbalanceadas? Como?

4. Como combinar diversos *rankings* de tal maneira a construir um *ranking* final com melhor desempenho que os *rankings* originais para o problema de ordenação binária? Como o desempenho desse *ranking* se compara com outras abordagens? Como utilizar *rankings* para derivar modelos de classificação?

## 1.3. Principais contribuições desta tese

As principais contribuições desta tese podem ser reunidas nos seguintes três grandes grupos:

1. novas abordagens para a geração de regras;

2. exploração de problemas relacionados à de proporções de exemplos entre as classes, principalmente proporções muito desbalanceadas dentro do contexto de aprendizado de máquina; e

3. um novo método para a combinação de *rankings* em problemas de ordenação binária

Todas as contribuições foram amplamente divulgadas à comunidade científica em diversos veículos.

No primeiro grupo destacam-se duas abordagens: na primeira delas, foi proposto um novo método para a geração de regras de exceções a partir de regras gerais (Prati et al., 2004e,d, 2003c,d,e, 2004c). Esse método está baseando em um princípio muito simples: primeiramente um algoritmo de aprendizado de máquina tradicional é utilizado para a indução de regras gerais; após, procuramos nas regras com altas taxas de erro por associações entre os exemplos incorretamente classificados e as classes diferentes daquela prevista pela regra geral; se associações fortes são encontradas, elas são consideradas como exceção à regra geral. Na segunda abordagem para geração de regras, foi proposto o algoritmo ROCCER (Prati & Flach, 2004, 2005), que faz seleção de regras baseado na análise ROC. O princípio por trás do algoritmo também é simples: uma nova regra é inserida somente se ela leva a um ponto que está fora da curva formada pela região convexa das regras que compõem o modelo. O algoritmo também possui outras qualidades, como um mecanismo implícito de *backtracking*. Em Batista et al. (2006) comparamos o desempenho do ROCCER com um algoritmo que também faz seleção de regras, mas utilizando um algoritmo genético cuja função de aptidão é a área abaixo da curva ROC (AUC).

Quanto à proporção de exemplos entre as classes, primeiramente exploramos a hipótese de que o grau de desproporção entre as classes não é o único fator para a queda de desempenho em conjuntos de dados com classes desbalanceadas, mas também participam o grau de sobreposição entre as classes e a quantidade absoluta de exemplos na classe minoritária (Prati et al., 2004a; Batista et al., 2005). Após, apresentamos resultados experimentais com vários métodos diferentes de balanceamento artificial de classes, dentre eles alguns por nós propostos (Batista et al., 2004; Prati et al., 2003a). Também reportamos resultados para averiguar o relacionamento entre classes desbalanceadas e pequenos disjuntos (Prati et al., 2004b). Um outro resultado obtido está relacionado com a sensitividade do algoritmo de aprendizado semi-supervisionado multi-visão Co-training à proporção de exemplos em cada classe que são rotulados em cada iteração do algoritmo (Matsubara et al., 2006).

Finalmente, desenvolvemos um método para a combinação de *rankings* em um *ranking* final para problemas de ordenação binária denominado BORDARANK. Esse método está baseado no método de votação por preferências borda count. Em linhas gerais, o método tem como entrada vários *rankings* e computa o *ranking* médio de cada caso. Essa abordagem, apesar de muito simples, mostrou-se competitiva com métodos mais sofisticados. Em Lee et al. (2006) exploramos a idéia de utilizar *rankings* para a avaliação de métodos de seleção de atributos. Como a avaliação de métodos de seleção de atributos é uma tarefa multi-objetivos, sendo que cada objetivo é geralmente medido em escalas diferentes (taxa de erro de classificação e porcentagem de atributos selecionados) a utilização de *rankings* provê uma maneira natural de combinar a avaliação desses objetivos. Além disso, a utilização de *rankings* permite uma fácil incorporação de outros objetivos, tais como complexidade sintática dos modelos induzidos.

Além desses, outros trabalhos relevantes foram publicados. Em Bernardini et al. (2005, 2006), propomos um método simbólico para a construção de *ensembles* de modelos de classificação. Nossa principal constatação nesse trabalho é que um *ensemble* construído a partir da melhor regra de um conjunto de regras, segundo alguns critérios de qualidade de regras pré-estabelecidos, tem um desempenho comparável, nos conjuntos de dados testados, a um *ensemble* construído levando-se em consideração todo o conjunto de regras. Essa constatação é importante no sentido de que as regras podem ser exploradas para explicar a decisão do *ensemble* em classificar um dado exemplo em uma certa classe, já que essa explicação é muito mais difícil levando-se em consideração todo o conjunto de regras.

Além das publicações científicas, também produzimos trabalhos de caráter mais didáticos. Em Monard & Prati (2005), publicamos um texto introdutório de aprendizado de máquina simbólico a partir de exemplos para mineração de dados. Em Prati et al. (2006), submetemos um artigo a respeito da análise ROC, no qual está baseado o Capítulo 4 desta tese.

## 1.4. Apresentação e resumo dos capítulos seguintes

Esta tese está organizada da seguinte maneira:

**Capítulo 2 — Aprendizado de máquina** Nesse capítulo é apresentada uma visão geral sobre a área de aprendizado de máquina. Para tanto, são apresentadas as características gerais dos sistemas de aprendizado, destacando os paradigmas, tarefas, modos e formas de aprendizado comumente utilizados nesses sistemas. É dada ênfase ao aprendizado

indutivo por exemplos que consiste em aprender conceitos a partir de exemplos e contra-exemplos desses conceitos.

**Capítulo 3 — Aprendizado de regras** Nesse capítulo são apresentadas algumas considerações a respeito de aprendizado de regras a partir de exemplos. São apresentadas as principais estratégias para aprendizado de regras tanto da família separar-para-conquistar quanto da família dividir-para-conquistar. Também são apresentadas algumas medidas de qualidade de regras.

**Capítulo 4 — Análise ROC** Nesse capítulo é feita uma introdução à análise ROC dentro do contexto de aprendizado de máquina. Análise ROC é discutidada como uma ferramenta extremamente útil para a avaliação de modelos induzidos a partir de algoritmos de aprendizado de máquina. Também é descrita a área abaixo da curva ROC como uma medida para a avaliação de modelos.

**Capítulo 5 — Novas abordagens para a geração de regras** Nesse capítulo são apresentadas as duas novas abordagens para a geração de regras propostas neste trabalho. Primeiramente é apresentado o método para geração de regras de exceção a partir de regras gerais; após, o algoritmo ROCCER baseado em análise ROC para a seleção de regras.

**Capítulo 6 — Classes desbalanceadas: entendeto o problema** Nesse capítulo é apresentada uma introdução ao problema de classes desbalanceadas. São apresentados diversos experimentos realizados com conjuntos de dados artificiais que tem como objetivo entender melhor o problema e confirmar nossa hipótese de que a sobreposição de exemplos entre as classes é um fator complicante ao problema de classes desbalanceadas.

**Capítulo 7 — Aprendizado e a proporção de exemplos entre as classes** Nesse capítulo, o problema de aprendizado com classes desbalanceadas é investigado em maior profundidade. São também apresentados experimentos em conjuntos de dados de *benchmark* utilizando vários métodos de balanceamento artificial de conjunto de dados, sendo que três deles foram propostos neste trabalho. Também são investigados outros problemas relacionados à proporção de exemplos entre as classes, tais como classes desbalanceadas e pequenos disjuntos, e com o algoritmo de aprendizado de máquina semi-supervisionado Co-training e a proporção de rotulamento de exemplos a cada iteração do algoritmo.

**Capítulo 8 — BordaRank: construindo *ensembles* de *rankings*** Nesse capítulo é apresentada uma nova abordagem para a combinação de *rankings* binários em um *ranking* final. Essa abordagem é inspirada no método de votação por preferências borda count.

**Capítulo 9 — Conclusão** Nesse capítulo são apresentadas as conclusões e limitações deste trabalho e propostas para trabalhos futuros.

## 1.5. Notas sobre tradução e terminologia

Alguns termos que aparecem no texto dessa tese, tais como *benchmark, marketing, ranking, backtracking, ensemble, shell, insights* e *kernel* não foram traduzidos para o portugues. Optamos por deixá-los em inglês pois é difícil encontrar uma tradução adequada que expresse o mesmo significado do temo em inglês. Outros termos foram traduzidos para o português, mas os acrônimos utilizados, tais como ILP, *tpr*, *fpr*, ROC, kNN, são os que correspondem aos termos em inglês. Esses acrônimos foram mantidas nessa forma por serem amplamente utilizados pela comunidade.

A utilização desses termos e acrônimos não representa de forma alguma desconsideração à lingua portuguesa, e somente foram utilizados pelos motivos acima apresentados.

Capítulo 2

# Aprendizado de máquina

Neste capítulo é feita uma introdução ao aprendizado de máquina, tema ao qual está relacionada esta tese. Ele está organizado da seguinte maneira: na Seção 2.1 são apresentadas algumas considerações iniciais sobre o tema. Na Seção 2.2 são apresentados os três métodos de inferência comumente utilizados em inteligência artificial. Aprendizado de máquina é contextualizado dentre esses métodos de inferência e na Seção 2.3 são apresentadas as principais características dos sistemas de aprendizado. Na Seção 2.4 é formalizado o problema de aprendizado. Finalmente, na Seção 2.5, são apresentadas as considerações finais deste capítulo.

## 2.1. Considerações iniciais

A construção de máquinas capazes de aprender por experiência tem sido, por um longo período, objeto de discussões tanto técnicas quanto filosóficas. O aspecto técnico desse debate vem recebendo grandes impulsos a partir do advento dos computadores eletrônicos. Relevantes pesquisas nessa área têm demonstrado que máquinas podem mostrar um significante nível de aprendizagem, mesmo não estando claramente definidas as fronteiras dessa habilidade de aprendizado.

A disponibilidade de sistemas de aprendizagem confiáveis é de suma importância, pois existem muitos problemas que não podem ser resolvidos por métodos clássicos de programação, uma vez que não é possível modelar esses problemas matematicamente ou algoritmicamente. Não se sabe, por exemplo, como escrever um programa de computador que reconheça caracteres escritos à mão. Entretanto, existe uma grande quantidade de exemplos disponíveis de caracteres escritos à mão e a sua representação eletrônica equivalente. É natural, de certa maneira, perguntar se é possível escrever um programa que aprenda por meio de exemplos a reconhecer um caractere manuscrito, uma vez que é de maneira semelhante que humanos aprendem a reconhecer o alfabeto.

O mesmo raciocínio pode ser aplicado à procura de genes em uma seqüência de DNA, filtragem de correspondência eletrônica, detecção ou reconhecimento de objetos em visão computacional, entre outros. Resolver algum desses problemas pode resultar em significativos avanços nas respectivas áreas e, se pudermos entender como programar computadores para resolvê-los, assim como para outros problemas de difícil solução e igual importância, poderemos obter resultados importantes na ciência da computação.

A solução dessas tarefas complicadas está claramente à margem das habilidades mesmo dos melhores programadores. Entretanto, o que pode ser feito é construir sistemas que, de certa maneira, tem a habilidade de aprender pela experiência. Neste capítulo são discutidos alguns aspectos relacionados ao aprendizado de máquina a partir de exemplos.

## 2.2. Inferência lógica e aprendizagem

A aprendizagem está relacionada à correta manipulação de conhecimento prévio e novas observações que possam levar a novos conhecimentos. Essa manipulação está relaciona a métodos de inferência lógica. Em geral, os métodos de inferência lógica podem ser agrupados em três classes, descritas a seguir.

**Dedução** Dedução poder ser definida como uma forma de inferência logicamente correta, isto é, a conclusão obtida a partir de um conjunto de premissas iniciais verdadeiras sempre preserva a verdade. Entretanto, a inferência dedutiva pode ser aplicada somente para obter informações que já estão contidas, de forma implícita, nas premissas.

Na aprendizagem por dedução, o aprendiz adquire conceitos por meio da inferência dedutiva sobre as premissas. Em outras palavras, essa abordagem inclui qualquer processo no qual o conhecimento apren-

dido é o resultado de transformações sobre o conhecimento existente, preservando a veracidade. Em geral, na aprendizagem por dedução, realiza-se uma seqüência de deduções e cálculos a respeito das informações presentes, visando a elicitação do conhecimento oculto, mas já existente nessas premissas.

**Abdução** Enquanto o resultado da inferência dedutiva não ultrapassa as premissas, em abdução estamos interessados em inferir um conhecimento particular a partir de observações e outras informações conhecidas. Possíveis hipóteses abdutivas são construídas a partir de um conjunto de predicados específicos não observados. Dessa maneira, hipóteses abdutivas são aquelas que trabalham com conhecimento incompleto, completando o conhecimento a respeito de um caso particular sobre um determinado indivíduo.

Em uma inferência abdutiva típica, assumimos que a descrição $D$ do domínio de um problema é suficiente, no sentido de que com a quantidade de informação disponível é possível aplicar inferências sobre ela. Na prática, isso significa que a parte incompleta da teoria pode ser isolada em alguns predicados não observados, que são chamados de predicados abdutíveis (ou abertos). Podemos então entender a teoria como a representação de todas as suas possíveis extensões abdutivas $T \cup \triangle$, para cada possível hipótese abdutiva $\triangle$. Uma enumeração de todas essas fórmulas (consistentes com a teoria $T$) fornecem o conjunto das possíveis extensões abdutivas de $T$. A implicação abdutiva sobre $T$ é então definida por implicações dedutivas em cada uma das extensões abdutivas. A veracidade da inferência está condicionada à veracidade das extensões.

**Indução** Indução é uma forma de inferência lógica que permite extrapolar as premissas, obtendo conclusões genéricas a partir de exemplos particulares. Nesse sentido, a indução pode ser caracterizada como uma forma de raciocínio que parte de um conceito específico e o generaliza, ou seja, da parte para o todo, do particular para o universal.

Ao contrário do que ocorre com a inferência dedutiva a partir de premissas válidas, o resultado da aplicação de inferência indutiva não preserva, necessariamente, a verdade, mesmo que o conjunto de premissas utilizado na inferência seja verdadeiro. Por esse motivo, o resultado da inferência indutiva é geralmente chamado de hipótese. Mesmo que não possamos garantir a validade de uma hipótese, pode-se atribuir a ela um certo grau de confiança, baseado na quantidade e qualidade das premissas, ou seja, quando as premissas utilizadas em uma

inferência indutiva são verdadeiras, e existe uma quantidade suficiente de premissas, é possível dizer que uma hipótese é provavelmente verdadeira.

O aspecto essencial da indução aplicada a aprendizado de máquina refere-se à chamada inferência amostra-para-população, exemplificada pelo seguinte modelo, usualmente chamado de generalização indutiva:

> Todos os objetos de uma amostra satisfazem a propriedade $\wp(x)$.
>
> Dessa maneira, todos os objetos na população satisfazem a propriedade $\wp(x)$.

Dessa forma, a inferência indutiva assume esse tipo de hipótese como válida e, similarmente a abdução, aplica inferência dedutiva para gerar novos conhecimentos. A diferença básica entre abdução e indução reside no fato de que essas hipóteses não estão limitadas a um subconjunto particular de predicados que são incompletamente especificados na representação do domínio do problema pela teoria $T$, mas elas são limitadas apenas pela própria linguagem da teoria $T$.

Ao se acrescentar novas premissas a um argumento dedutivo válido, não se afeta a validade do argumento. No entanto, no caso de inferência indutiva, ao se acrescentar premissas adicionais pode ocorrer uma mudança no grau de confiança do argumento. Nesse caso, a adição de uma nova premissa pode aumentar ou diminuir a confiança de um argumento indutivo, ou até mesmo invalidá-lo.

Dedução é um dos métodos mais utilizados em IA. Resolução, por exemplo, é um dos mecanismos de inferência lógica mais utilizados em *shells* de sistemas especialistas e é a base do motor de inferência da linguagem de programação lógica PROLOG. Entretanto, dada uma teoria incompleta (ou mesmo uma teoria vazia como em muitas aplicações de mineração de dados) existem ocasiões nas quais a inferência dedutiva não é suficiente para resolver um problema. Notadamente, podemos destacar duas classes de tarefas:

a. a procura por qual porção de informação pode ser verdadeira de acordo com uma teoria geral que descreve um domínio de interesse, e

b. a construção de novas teorias que descrevem o comportamento presente de um determinado processo e possam prever o seu comportamento futuro.

Claramente, para essas duas classes de problemas, inferência não-dedutiva é necessária para suprir a necessidade de explicação (a) e generalização (b). Devido ao seu inerente poder de explicação (utilizando uma teoria $T$ e assumindo uma suposição $\triangle$), abdução é freqüentemente utilizada em diagnóstico e planejamento. Em compensação, o processo de indução é um dos principais meios para se inferir novos conhecimentos e, por esse motivo, indispensável no processo de aquisição e descoberta de conhecimento. Nesse sentido, indução é uma das formas de inferência mais utilizadas em aprendizado de máquina.

## 2.3. Características gerais dos sistemas de AM

Um sistema de aprendizagem é um programa de computador que toma decisões baseadas em experiências acumuladas por meio da solução de problemas anteriores. Os sistemas de AM possuem características particulares e comuns que possibilitam uma certa classificação quanto à linguagem de descrição, modo, paradigma, tarefa e forma de aprendizagem utilizadas nesses sistemas, resumidos na Tabela 2.1 e sucintamente descritos a seguir.

| *Modos de Aprendizado* | *Paradigmas de Aprendizado* | *Linguagens de Descrição* | *Formas de Aprendizado* | Tarefas de aprendizado |
|---|---|---|---|---|
| Supervisionado | Simbólico | $\mathcal{LD}_{\mathcal{E}}$ - Exemplos ou Objetos | Incremental | Classificação |
| Não Supervisionado | Estatístico | $\mathcal{LD}_{\mathcal{H}}$ - Hipóteses ou Conceitos Aprendidos | | |
| Semi Supervisionado | Baseado em protótipos | $\mathcal{LD}_{\mathcal{C}}$ - Teoria de Domínio ou Conhecimento de Fundo | Não Incremental | Ordenação |
| Por reforço | Conexionista | | | Regressão |
| | Genético | | | |

**Tabela 2.1:** Características gerais de sistemas de aprendizado de máquina

### 2.3.1 Modos de aprendizagem

Segundo (Russell & Norvig, 2003), para alguns sistemas de aprendizagem é necessário predizer se uma certa ação irá fornecer uma certa saída. Nesse sentido, é possível classificar os sistemas de AM nos seguintes modos:

***Aprendizagem supervisionada,*** no qual dado um conjunto de observações ou exemplos rotulados, isto é, conjunto de observações em que a classe, denominada também atributo meta, de cada exemplo é conhecida, o objetivo é encontrar uma hipótese capaz de classificar novas observações entre as classes já existentes.

***Aprendizagem não-supervisionado,*** no qual dado um conjunto de observações ou exemplos não rotulados, o objetivo é tentar estabelecer a existência de grupos ou similaridades nesses exemplos.

***Aprendizagem semi-supervisionado,*** no qual dado um pequeno conjunto de observações ou exemplos rotulados e um conjunto de observações ou exemplos não rotulados, o objetivo é utilizar ambos os conjuntos para encontrar uma hipótese capaz de classificar novas observações entre as classes já existentes. Aprendizagem semi-supervisionada é um meio termo entre aprendizagem supervisionada e não-supervisionada.

***Aprendizagem por reforço,*** no qual o agente aprendiz interage com o meio ambiente que o cerca e aprende uma política ótima de ação por experimentação direta com o meio. Dependendo de suas ações, o aprendiz é recompensado ou penalizado. O objetivo do aprendiz é desenvolver uma política ótima que maximize a quantidade de recompensa recebida ao longo da sua execução.

### 2.3.2 Paradigmas de aprendizagem

Atualmente, já foram propostos diversos paradigmas de AM. Nesta seção são apresentados brevemente alguns deles, tais como o paradigma simbólico, estatístico, baseado em protótipo, conexionista e genético.

#### Paradigma simbólico

Os sistemas de aprendizagem simbólica buscam aprender construindo representações simbólicas de um conceito por meio da análise de exemplos e contra-exemplos desse conceito. As representações simbólicas estão tipicamente representadas na forma de alguma expressão lógica, árvore de decisão, regras de produção ou rede semântica.

Atualmente, entre as representações simbólicas mais estudadas estão as árvores e regras de decisão. Métodos de indução de árvores de decisão a partir de dados empíricos, conhecidos como particionamento recursivo, foram estudados por pesquisadores da área de Inteligência Artificial e Estatística. Os sistemas ID3 (Quinlan, 1986) e C4 (Quinlan, 1987) para indução de árvores de decisão tiveram uma importante contribuição sobre a pesquisa em IA. O sistema de classificação de árvores de regressão CART (Breiman et al., 1984) foi desenvolvido por estatísticos, durante praticamente o mesmo período que o ID3, no final dos anos 70.

Os trabalhos com indução de regras de decisão surgiram com a família de algoritmos AQ (Michalski, 1969), e precederam algoritmos como o

CN2 (Clark & Niblett, 1989; Clark & Boswell, 1991). Uma outra abordagem surgiu da simples tradução das árvores de decisão para regras, com um posterior refinamento (Quinlan, 1987), resultando no algoritmo C4.5rules (Quinlan, 1993). Aprendizado simbólico de regras é discutido em maiores detalhes no Capítulo 3.

### Paradigma estatístico

Pesquisadores em estatística têm criado diversos métodos de classificação, muitos deles semelhantes aos métodos empregados pela comunidade científica em aprendizado de máquina. Por exemplo, o método CART (Breiman et al., 1984), mencionado anteriormente, é um sistema muito conhecido para construir árvores de decisão, desenvolvido por estatísticos. Como regra geral, técnicas estatísticas tendem a focar tarefas em que todos os atributos têm valores contínuos ou ordinais. Muitos desses métodos também são paramétricos, assumindo algum modelo pré-estabelecido, e então ajustando valores apropriados para os parâmetros do modelo a partir dos dados. Por exemplo, um modelo de classificação linear assume que as classes podem ser expressas como combinação linear dos valores dos atributos, e então procura uma combinação linear particular que fornece a melhor aproximação sobre o conjunto de dados. Os modelos estatísticos freqüentemente assumem que os valores de atributos estão normalmente distribuídos, e então usam os dados fornecidos para determinar média, variância e co-variância da distribuição. Alguns autores têm considerado redes neurais como métodos estatísticos paramétricos, uma vez que treinar uma rede neural geralmente significa encontrar valores apropriados para pesos pré-determinados.

### Paradigma baseado em protótipo

Uma forma de classificar um caso é lembrar de um caso similar cuja classe é conhecida e assumir que o novo caso terá a mesma classe. Essa filosofia exemplifica os sistemas baseados em protótipos, que classificam casos nunca vistos utilizando casos similares conhecidos.

Em sua forma mais simples, sistemas que empregam esse paradigma armazenam todos os exemplos de treinamento. A classificação é dada pela maior quantidade de exemplos vizinhos de uma dada classe. Essa abordagem é conhecida como $k$-vizinhos-mais-próximos (kNN, do inglês *k-nearest-neighbours*). Outras abordagens empregam heurísticas para selecionar os exemplos armazenados. Saber quais casos de treinamento devem ser memorizados é importante para evitar dificuldades e lentidão de manuseio por parte do modelo de classificação. O ideal é reter apenas os casos com os

quais seja possível resumir toda a informação. Em Aha et al. (1991) estão descritas algumas estratégias para decidir quando um novo caso deve ser memorizado. A medida de similaridade para os casos nos quais todos os atributos são contínuos pode ser calculada por meio de alguma distância entre esses atributos. Na presença de atributos ordinais essa medida se torna complicada, bem como na presença de atributos irrelevantes, os quais podem fazer com que dois casos similares sejam interpretados como muito diferentes. Métodos sensíveis ao contexto, que alterem a escala dos atributos, podem melhorar estas medidas (Stanfill & Waltz, 1986).

Raciocínio baseado em casos é uma outra família de algoritmos de aprendizado que se enquadra nesse paradigma. Nessa abordagem, casos protótipos são construídos e exemplos são classificados com base na similaridade com esses casos protótipos.

### Paradigma conexionista

Redes neurais são construções matemáticas relativamente simples, que foram inspiradas em modelos biológicos do sistema nervoso. A representação de uma rede neural envolve unidades altamente interconectadas e, por esse motivo, o nome conexionismo é utilizado para descrevê-las.

As pesquisas em redes neurais foram iniciadas com o trabalho pioneiro de McCulloch & Pitts (1943). McCulloch era um psiquiatra e pesquisou por 20 anos uma forma de representar um evento no sistema nervoso. Pitts era um jovem pesquisador e começou a trabalhar com McCulloch em 1942. Praticamente 15 anos após a publicação de McCulloch e Pitts, Rosenblatt (1962) apresentou o perceptron, cuja grande contribuição foi a prova do teorema de convergência. Mas Minsky & Papert (1969) demonstraram a existência de limites fundamentais nos perceptrons de apenas uma camada.

A pesquisa na área ficou praticamente estática até que Hopfield (1982) utilizou a idéia de uma função de energia para formular uma nova forma de compreender os cálculos realizados em redes recorrentes com conexões sinápticas simétricas. O artigo de Hopfield em 1982 e o livro de Rumelhart & McClelland (1986) foram às publicações que mais influenciaram no ressurgimento do interesse sobre redes neurais na década de 80. Redes neurais tiveram um longo caminho desde McCulloch e Pitts, e continuam a crescer em teoria, projetos e aplicações (Braga et al., 2003).

A metáfora biológica com as conexões neurais do sistema nervoso tem interessado muitos pesquisadores e tem subsidiado discussões sobre os méritos e as limitações dessa abordagem de aprendizagem. Em particular, as analogias com a biologia têm levado muitos pesquisadores a acreditar que as redes neurais possuem um grande potencial na resolução de pro-

blemas que requerem intenso processamento sensorial humano, tal como visão e reconhecimento de voz e imagens.

#### Paradigma genético

Este formalismo de classificação é derivado do modelo evolucionário de aprendizagem (Holland, 1975). Um modelo de classificação genético consiste de uma população de elementos de classificação que competem para fazer a predição. Elementos que possuem um desempenho fraco são descartados, enquanto os elementos mais fortes proliferam, produzindo variações de si mesmos. Este paradigma possui uma analogia direta com a teoria de Darwin, na qual sobrevivem os mais bem adaptados ao ambiente.

Alguns operadores genéticos básicos que aplicados a uma população geram novos indivíduos são *Reprodução, Cruzamento, Mutação e Inversão*. Esses operadores atuam no controle da quantidade de cópias produzidas de um indivíduo, na troca de material genético, na preservação de uma espécie e na manutenção de uma certa diversidade na nova população.

### 2.3.3 Linguagens de descrição

Qualquer que seja o tipo de aprendizagem é necessário uma linguagem para descrever objetos (ou possíveis eventos), uma linguagem para descrever conceitos, ou hipóteses, e uma linguagem para descrever conhecimento de fundo. AM utiliza vários tipos de linguagem de descrição, entre elas representações equivalentes à lógica proposicional e relacional, agrupamentos, hierarquias de conceitos e redes probabilísticas. Mais detalhes sobre linguagens de descrição são apresentados na Seção 3.1.

### 2.3.4 Aprendizagem incremental e não-incremental

Os algoritmos de aprendizagem podem ser classificados de duas maneiras, segundo o modo em que os exemplos são apresentados

**não incremental** que necessita que todos os exemplos estejam disponíveis simultaneamente para que seja induzido um conceito. É vantajoso usar esses algoritmos para problemas de aprendizagem nos quais todos os exemplos estão disponíveis e, provavelmente, não irão ocorrer mudanças.

**incremental** que revê a definição do conceito corrente, se necessário, em resposta a cada novo exemplo observado. Os exemplos observados são considerados um a um pelo sistema, isto é, o sistema considera o

primeiro exemplo e, de acordo com esse exemplo constrói uma determinada hipótese; a seguir considera um segundo exemplo, que pode ou não modificar a primeira hipótese, baseando-se em como ela classifica o segundo exemplo. Dessa forma, o sistema continua modificando o conceito à medida que mais exemplos são a ele apresentados.

Uma das vantagens de usar um algoritmo incremental é que o conhecimento pode ser rapidamente atualizado a cada nova observação. Porém, eventualmente, pode ser mais eficiente revisar uma hipótese existente do que gerar uma nova hipótese cada vez que um novo exemplo é observado (Utgoff, 1989).

### 2.3.5 Tarefas de aprendizado

Em geral, em problemas de aprendizado supervisionado, cada exemplo é descrito por um vetor de valores de características e por um atributo especial que descreve uma característica de interesse na qual estamos interessados em criar o modelo. Esse atributo pode ser discreto, ordinal ou contínuo. No caso do atributo discreto, o problema é conhecido como problema de *classificação*, e o objetivo é classificar futuros casos em cada uma das classes pré-estabelecidas. Caso o atributo seja contínuo, o problema é geralmente conhecido como problema de *regressão*, e o objetivo é prever o valor desse atributo com base nas características dos exemplos. No caso do atributo meta ser ordinal, o problema é conhecido como *ordenação* ou *regressão logística*, e o objetivo é ordenar um conjunto de casos de acordo com uma característica de interesse.

Note que mesmo que os problemas sejam definidos de acordo com o tipo do atributo meta, é possível utilizar variáveis de outros tipos para cumprir a tarefa. Por exemplo, é possível discretizar um atributo meta contínuo e prever uma faixa de valores (o atributo discretizado) ao invés de um valor contínuo. Também é possível prever um valor contínuo para um problema com atributo meta discreto. Nesse caso, cada classe pode ser associada a uma faixa de valores contínuos. Além disso, é possível "calibrar" essas faixas, para melhorar o desempenho. No capítulo 8, essa idéia é aplicada na ordenação de exemplos em problemas de classificação (ordenação binária).

## 2.4. Aprendizado de máquina indutivo por exemplos

O objetivo principal das pesquisas em AM é construir máquinas capazes de aprender por experiência. Neste trabalho, utilizaremos a definição de

AM dada por Mitchell (1997a), que inclui qualquer programa de aprendizagem que melhora o seu desempenho em uma dada tarefa utilizando alguma experiência. Mais precisamente:

> "Diz-se que um programa de computador aprende a partir da experiência $\mathcal{E}$ com respeito a algumas classes de tarefas $\mathcal{T}$ e uma dada medida de desempenho $\mathcal{P}$ se o seu desempenho nas tarefas $\mathcal{T}$, medidas por $\mathcal{P}$, melhoram com a experiência $\mathcal{E}$"

Por exemplo, um programa de computador capaz de aprender a jogar xadrez pode ter como medida de desempenho a sua habilidade em vencer no que se refere à classe de tarefas jogar xadrez, utilizando como experiência jogos de xadrez contra si próprio.

A questão da aprendizagem é essencial em IA, uma vez que ser ou não capaz de aprender é uma habilidade essencial para um sistema apresentar qualquer comportamento inteligente. Em AM, estudam-se métodos computacionais capazes de melhorar o seu desempenho pela aquisição de novos conhecimentos, novas habilidades e novos meios de organizar o conhecimento já existente. O estudo de métodos de aprendizagem pode levar a um melhor entendimento da nossa própria inteligência e do nosso excepcional processo de aprendizagem, inferência, adaptação e indução (Schapire, 2001).

Em AM, estuda-se como modelar o processo de aprendizagem. Em qualquer processo de aprendizagem, o aprendiz deve utilizar os conhecimentos que possui pra obter novos conhecimentos. Neste trabalho nos concentramos em aprendizagem simbólico supervisionado utilizando classificação e ordenação binária. O termo simbólico indica que os modelos induzidos devem ser legíveis e interpretáveis por humanos. O termo supervisionado sugere que algum processo, algumas vezes denominado agente externo ou professor, previamente rotulou os dados disponíveis para a aprendizagem. Finalmente, o termo classificação denota o fato que o conceito meta (o atributo rotulado) a ser aprendido é discreto, ou seja, ele consiste de valores nominais, e o termo ordenação binária significa que queremos ordenar os exemplos da classe de interesse à frente dos outros exemplos.

Na aprendizagem supervisionada por exemplos, geralmente, o aprendiz induz uma hipótese $\mathcal{H}$ que descreve um conceito $\mathcal{C}$ a partir de um conjunto de exemplos e contra-exemplos $\mathcal{E}$. Por exemplo, o conceito "é verde" divide o mundo em todos os objetos que são e os que não são verdes. Ao algoritmo de aprendizagem são apresentados exemplos do conceito, e, para cada exemplo, é dito se ele é um exemplo positivo ou negativo do conceito.

Ao universo de objetos para os quais são apresentados exemplos ao aprendiz é chamado de domínio $\mathcal{D}$ (ou espaço de descrição dos exemplos)

e cada objeto do domínio representa um exemplo. No caso do conceito "é verde", o domínio pode consistir de todas as frutas do planeta, e os exemplos apresentados ao aprendiz seriam a descrição de algumas frutas conhecidas. Esse descrição é normalmente realizada utilizando um conjunto de atributos que descrevem características particulares de cada exemplo. No caso das frutas, haveriam atributos como tamanho, forma, origem, quantidade de sementes, etc. Dentre esses atributos, o atributo meta (aprendizagem supervisionada) distingue os exemplos positivos dos negativos. No caso das frutas, o atributo meta seria a cor da casca. Também podem ser fornecidos ao aprendiz algum conhecimento prévio a respeito do domínio, com o objetivo de guiar/direcionar a indução do conceito meta. Por exemplo, poderia ser dada informação que algumas frutas, antes de amadurecerem são verdes, mas que posteriormente mudam a cor. A tarefa do aprendiz é induzir uma hipótese capaz de distinguir os exemplos positivos (valor do atributo meta "é verde") dos negativos. Bratko (2001) formaliza o problema de aprendizagem como:

> Seja $\mathcal{U}$ o conjunto universal dos objetos, isto é, todos os objetos que o aprendiz pode encontrar. Não existe limites, a princípio, para o número de exemplos de $\mathcal{U}$. Um conceito $\mathcal{C}$ pode ser formalizado como sendo um subconjunto de objetos de $\mathcal{U}$, ou seja, $\mathcal{C} \subset \mathcal{U}$. Aprender um conceito $\mathcal{C}$ significa aprender a reconhecer objetos em $\mathcal{C}$. Ou seja, uma vez que o conceito $\mathcal{C}$ é aprendido, para qualquer objeto $x \in \mathcal{U}$, o sistema deve ser capaz de reconhecer se $x \in \mathcal{C}$.

Esse descrição pode ser estendida para problemas de mais de duas classes (conhecidos como problemas multi-classe) ou para atributos metas contínuos ou ordenação. No caso de problemas multi-classe, pode-se relaxar a definição do conceito $\mathcal{C}$ para representar um conjunto discreto de $n_{cl}$ classes $C \in \{c_1, c_2, \ldots, c_{n_{cl}}\}$, de tal maneira que o objetivo é reconhecer se um objeto $x$ pertence a uma das $n_{cl}$ classes. No caso de regressão, relaxamos mais uma vez a definição do conceito $\mathcal{C}$ para representar um valor numérico $C \in \mathbb{R}$, e o objetivo é predizer esse valor numérico. Da mesma maneira, no caso de ordenação, relaxamos a definição do conceito $\mathcal{C}$ para um valor numérido $C \in \mathbb{N}$. Considerando essas extensões, podemos redefinir o problema de aprendizagem como:

> Seja $D$ o domínio do conjunto universal de objetos $\mathcal{U}$ e $C$ o domínio do conceito $\mathcal{C}$ a ser aprendido. O objetivo da aprendizagem é encontrar uma função $\mathrm{H} : D \rightarrow C$, que represente uma hipótese $\mathcal{H}$, que aproxime a função $\mathrm{F}$ (desconhecida) que mapeia $\mathcal{U}$ em $\mathcal{C}$, ou seja, $\mathrm{F} : D \rightarrow C$.

Para determinar se o aprendiz teve sucesso em "aprender" o conceito, pode ser feito um teste: é apresentado um conjunto de exemplos cujos valores do atributo meta são conhecidos, mas omitidos para fins de avaliação, e verifica-se o quão apropriadamente a hipótese $\mathcal{H}$ aproxima $\mathrm{F}$.

Um outro fator importante que deve ser considerado quanto à avaliação de hipóteses é que elas devem fornecer uma descrição mais compacta do conceito embutido nos exemplos. Ou seja, supondo um conjunto de exemplos com cardinalidade $M$, espera-se que o modelo induzido tenha uma descrição menor do que $M$, pois, caso contrário, os dados descreveriam melhor a si próprios e de forma mais compacta. O tamanho do modelo influencia no quão compreensível é um modelo para os humanos.

Além do tamanho do modelo, um outro fator que influencia na compreensão de modelos é a linguagem de descrição das hipóteses. De acordo com (Michalski, 1983), modelos de classificação podem ser agrupados em duas grandes categorias:

1. sistemas caixa-preta, que desenvolvem sua própria representação do conceito, isto é, sua representação interna pode não ser facilmente interpretada por humanos e não fornecem nem esclarecimento nem explicação do processo de reconhecimento de novos exemplos;

2. sistemas orientados a conhecimento, que objetivam a criação de estruturas simbólicas que sejam compreensíveis por humanos.

Na realidade, uma distinção interessante dessas duas categorias foi formulada por Michie (1988) em termos de três critérios:

1. **critério fraco**: o sistema usa exemplos para manter uma base atualizada para a melhoria do desempenho em exemplos subseqüentes. Métodos baseado em protótipos satisfazem este critério;

2. **critério forte**: o critério fraco é satisfeito. Além disso, o sistema é capaz de comunicar sua representação interna em forma simbólica explícita;

3. **critério ultra-forte**: os critérios fraco e forte são satisfeitos. Além disso, o sistema é capaz de comunicar sua representação interna em forma simbólica explícita, a qual pode ser usada por um humano sem a ajuda de um computador, ou seja, utilizando apenas a mente humana.

Neste trabalho nos focalizamos na aprendizagem de conceitos, cujo interesse principal consiste em obter descrições simbólicas que sejam facilmente compreendidas por seres humanos, utilizando apenas modelos mentais. Mais precisamente, nos concentraremos no estudo de sistemas orientados a conhecimento e que satisfazem os critérios forte ou ultra-forte,

ou seja, sistemas de aprendizagem que aprendem no nível de conhecimento (Dietterich, 1990).

## 2.5. Considerações finais

Inferência lógica é um dos recursos mais utilizados em IA para manipular conhecimento. Inferência indutiva é capaz de a partir de um conjunto de observações que especificam um conhecimento incompleto (geralmente extensional) sobre as observações, generalizá-las em novos conhecimentos a respeito das observações. Por esse motivo, inferência indutiva é largamente utilizada em aprendizado de máquina.

Neste capítulo foi apresentada uma breve introdução ao aprendizado de máquina a partir de exemplos. Primeiramente foram apresentadas algumas considerações sobre inferência lógica e aprendizado de máquina. Também foram apresentadas as principais características de sistemas de aprendizado. No próximo capítulo é apresentado em maiores detalhes a indução de regras a partir de exemplos.

Capítulo 3

# Indução de regras

UMA das contribuições desta tese está relacionada à indução de regras, descritas no Capítulo 5. Para fornecer ao leitor um plano de fundo à leitura desta tese, neste capítulo é apresentada uma introdução à indução de regras. Na Seção 3.2, aprendizado de regras é formalizado como um problema de busca. Nas Seções 3.3 e 3.4 são apresentados, respectivamente, o aprendizado de uma única regra e de um conjunto de regras. Na Seção 3.5 é apresentada a indução de árvores de decisão. Na Seção 3.6 são descritas algumas medidas de avaliação de regras. Finalmente, na Seção 3.7 são apresentadas as considerações finais do capítulo.

## 3.1. Linguagens de descrição

Qualquer que seja o tipo de aprendizado, é necessária uma maneira de descrever exemplos, modelos e conhecimento do domínio. Para descrevê-los, as seguintes linguagens de representação (ou linguagens de descrição são usadas):

- Linguagens de descrição de exemplos;
- Linguagens de descrição de hipóteses;

- Linguagens de descrição de conhecimento do domínio.

A seguir são descritas algumas linguagens de representação freqüentemente utilizadas em AM simbólico em ordem crescente de complexidade e força expressiva. No caso de AM simbólico, as linguagens de descrição mais freqüentemente utilizadas, em ordem crescente de complexidade e força expressiva, são: de ordem zero, baseada em atributos e baseada em lógica de primeira ordem.

**Lógica de Ordem Zero ou Proposicional** Na lógica de ordem zero ou cálculo proposicional, o item a ser representado é descrito por conjunções, disjunções e negações de constantes booleanas que representam atributos individuais. Por exemplo:

$$\text{fêmea} \wedge \text{adulta} \rightarrow \text{pode\_ter\_filhos}$$

Esta linguagem tem um baixo poder descritivo, não sendo capaz de descrever objetos sobre os quais relações são observadas.

**Lógica de Atributos** De forma a representar itens, vários algoritmos proposicionais utilizam uma linguagem baseada em atributos (Michalski, 2004). Formalmente, a lógica de atributos é equivalente à lógica proposicional, mas emprega uma notação mais poderosa e flexível. Essa forma de notação é comumente conhecida como formato atributo-valor. A melhoria ocorre pois os atributos são tratados como variáveis que podem assumir diversos valores. Por exemplo:

$$\text{sexo}{=}\text{feminino} \wedge \text{idade}{=}\text{adulta} \rightarrow \text{classe}{=}\text{pode\_ter\_filhos}$$

Embora a maioria dos algoritmos de AM utilize a lógica de atributos para descrever exemplos e hipóteses, sua baixa capacidade de expressão impede a representação de objetos estruturados, assim como as relações entre objetos ou entre seus componentes. Dessa maneira, aspectos relevantes dos exemplos que, de alguma maneira poderiam caracterizar o conceito sendo aprendido, podem não ser representados.

**Lógica de Primeira Ordem** Para superar as limitações de representação impostas por uma linguagem de atributos, o aprendizado utilizando representações que possuem maior poder, tais como algumas variações da lógica de primeira ordem, tem recebido maior atenção. A lógica

de primeira ordem permite descrever e raciocinar sobre objetos e predicados que especificam propriedades de objetos ou relacionamentos entre objetos do domínio.

Um subconjunto importante da lógica de primeira ordem é composto pelas cláusulas de Horn. Uma cláusula de Horn consiste em uma regra cuja cabeça contém um único predicado e um corpo com zero, um ou mais predicados. O seguinte exemplo, na sintaxe proposta por (Kowalsky, 1979) para a linguagem de programação lógica PROLOG, descreve que uma pessoa $X$ é irmão da pessoa $Y$ se $X$ é homem e ambos $X$ e $Y$ possuem o mesmo pai $Z$, na qual $X$, $Y$ e $Z$ são variáveis que representam objetos.

$$\text{irmão}(X,Y) \texttt{ :- } \text{homem}(X), \text{pai}(Z,X), \text{pai}(Z,Y).$$

O elemento à esquerda do símbolo `:-` é a cabeça e o que está direita do símbolo é o corpo (ou cauda) da cláusula. O símbolo `:-` é equivalente à implicação lógica $\leftarrow$ e é denominado *neck*[1]. As vírgulas separando cada predicado significam conjunções lógicas. Além disso, todas as variáveis estão sempre universalmente quantificadas, ou seja, no exemplo acima, a cláusula é verdadeira para todo $X, Y, Z \in \mathcal{D}$. As variáveis entre parênteses são chamadas de argumentos.

Nota-se que se todos os predicados não possuem argumentos, a linguagem se reduz à lógica de ordem zero e se todos os predicados possuem um único argumento constante (sem variáveis envolvidas), a linguagem se reduz à lógica de atributos.

Neste trabalho trataremos de algoritmos que usam linguagens baseadas em atributos para descrever os exemplos e as hipóteses por eles induzidas. Deve ser observado que apesar da maioria dos sistemas de aprendizado utilizarem lógica de atributos, eles são limitados devido à pouca expressividade do formalismo representacional e a capacidade limitada de utilizar conhecimento do domínio. Em contrapartida, algoritmos simbólicos que usam linguagens baseadas em lógica de primeira ordem ou superiores, chamados de sistema de aprendizado relacional, dentre os quais destacam-se os algoritmos de Programação Lógica Indutiva (Lavrač & Džeroski, 1994) (ILP), possuem uma alta expressividade para representar conceitos e a habilidade de representar conhecimento do domínio. Entretanto, eles são mais complexos e exigem um alto poder computacional para a resolução de problemas reais.

[1] $q$ `:-` $p \equiv q \leftarrow p \equiv p \rightarrow q$

### 3.1.1 Representação dos dados

Um algoritmo para a indução de regras de classificação recebe como entrada um conjunto de casos ou exemplos cuja classificação é conhecida. Um caso é descrito por um conjunto fixo de atributos: $A_i$, $i \in \{1, \ldots, n_{atr}\}$. Um atributo pode tanto assumir um conjunto finito de valores (atributo discreto ou qualitativo) ou um número real (atributo contínuo ou quantitativo). Um exemplo $e_j$ é um vetor de valores de atributos rotulado com a sua respectiva classe, ou seja, $e_j = (v_{1,j}, \ldots, v_{n_{atr},j}, c_{i,j})$, no qual cada $v_{i,j}$ é um possível valor do atributo $A_i$ e $c_{i,j}$ é uma dos $n_{cl}$ possíveis valores do atributo classe. Um conjunto de dados de cardinalidade $n_{ex}$ é um conjunto contendo $n_{ex}$ exemplos.

| | $A_1$ | $A_2$ | $\ldots$ | $A_{n_{atr}}$ | $C$ |
|---|---|---|---|---|---|
| $e_1$ | $x_{1,1}$ | $x_{1,2}$ | $\ldots$ | $x_{1,n_{atr}}$ | $c_{i,1}$ |
| $e_2$ | $x_{2,1}$ | $x_{2,2}$ | $\ldots$ | $x_{2,n_{atr}}$ | $c_{i,2}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ |
| $e_{n_{ex}}$ | $x_{n_{ex},1}$ | $x_{n_{ex},2}$ | $\ldots$ | $x_{n_{ex},n_{atr}}$ | $c_{i,n_{ex}}$ |

**Tabela 3.1:** Conjunto de exemplos no formato atributo-valor

Para induzir um modelo de classificação, o algoritmo de aprendizado supervisionado utiliza uma amostra de exemplos ou casos para os quais se conhece a classificação verdadeira. Cada caso é descrito por um conjunto de atributos. Para se distinguir casos entre as possíveis classificações, cada caso é rotulado com um atributo especial, denominado classe, cujos valores se referem à classificação verdadeira dos casos. Casos rotulados são chamados de exemplos, e a amostra é geralmente chamada de conjunto de exemplos de treinamento.

Suponha que se tenha um conjunto de dados de registros de pacientes com os respectivos diagnósticos, com o qual queiramos induzir um conjunto de regras para um diagnóstico ou prognóstico. Neste tipo de problema, é comum que cada registro de paciente seja descrito por alguns atributos que podem ser contínuos (por exemplo a idade) ou discretos (por exemplo o sexo do paciente, que pode ser masculino ou feminino). Na Tabela 3.2 estão contidas informações simplificadas do problema de prescrição de lentes de contato (Witten & Frank, 2000; Flach & Lavrač, 2003). Nessa tabela, os pacientes são descritos por quatro atributos: Idade, tipo de lentes prescritas (Espectropia), Astigmatismo e taxa de produção de lágrimas (Produção lacrimal). Os registros dos pacientes são rotulados com três possíveis rótulos de classes, denotando o tipo de lentes de contato prescritas a um dado indivìduo: $C$ = {nenhuma, macia, dura}.

| Idade | Espectropia | Astigmatismo | Produção lacrimal | Lentes |
|---|---|---|---|---|
| jovem | miopia | não | reduzido | nenhuma |
| jovem | miopia | não | normal | macia |
| jovem | miopia | sim | reduzido | nenhuma |
| jovem | miopia | sim | normal | dura |
| jovem | hipermetropia | não | reduzido | nenhuma |
| jovem | hipermetropia | não | normal | macia |
| jovem | hipermetropia | sim | reduzido | nenhuma |
| jovem | hipermetropia | sim | normal | dura |
| pre-presbiotico | miopia | não | reduzido | nenhuma |
| pre-presbiotico | miopia | não | normal | macia |
| pre-presbiotico | miopia | sim | reduzido | nenhuma |
| pre-presbiotico | miopia | sim | normal | dura |
| pre-presbiotico | hipermetropia | não | reduzido | nenhuma |
| pre-presbiotico | hipermetropia | não | normal | macia |
| pre-presbiotico | hipermetropia | sim | reduzido | nenhuma |
| pre-presbiotico | hipermetropia | sim | normal | nenhuma |
| presbiotico | miopia | não | reduzido | nenhuma |
| presbiotico | miopia | não | normal | nenhuma |
| presbiotico | miopia | sim | reduzido | nenhuma |
| presbiotico | miopia | sim | normal | dura |
| presbiotico | hipermetropia | não | reduzido | nenhuma |
| presbiotico | hipermetropia | não | normal | macia |
| presbiotico | hipermetropia | sim | reduzido | nenhuma |
| presbiotico | hipermetropia | sim | normal | nenhuma |

**Tabela 3.2:** O conjunto de dados de lentes de contato

É importante observar que este conjunto de dados é completo, no sentido de que todas as possíveis combinações de valores de atributos estão presentes. Neste caso, o algoritmo de aprendizagem não pode generalizar além da classificação que já está presente no conjunto de treinamento. Na prática, o objetivo da aprendizagem neste caso pode ser entendido como a transformação dos dados em uma forma mais compacta, sem perder o poder de classificação.

Além disso, deve ser observado que todos os quatro atributos da Tabela 3.2 possuem valores discretos. Para três dos quatro atributos, Spectropia, Astigamatismo e Produção lacrimal, esta é a representação mais natural. Por sua vez, Idade seria naturalmente melhor representado por um atributo numérico. Por simplicidade, esse atributo foi discretizado em três valores: jovem, pre-presbiotico e presbiotico. Essa discretização foi feita por um especialista, para indicar as categorias de idade que se encaixam os pacientes com características específicas. Esses valores também poderiam ser discretizados em intervalos tanto no processamento dos dados ou no próprio processo de indução de regras. A discretização de atributos é

uma caracteristica muito importante na representação do conhecimento, mas foge do escopo deste trabalho.

### 3.1.2 Linguagem atributo-valor para regras if-then

Dado um conjunto de exemplos classificados, um algoritmo de aprendizagem de regras constrói um conjunto de regras do tipo if-then . Uma regra if-then tem o formato:

**if** *Condições* **then** *Conclusão*.

*Condições* contém uma ou mais restrições quanto a valores dos atributos, ***i.e.***, condições da forma $A_i = v_{ij}$ para os atributos discretos e $A_i < v$ ou $A_i \geq v$ para atributos contínuos (no qual $v$ é um valor limiar que não corresponde, necessariamente, a um valor de um atributo observado nos exemplos) Flach & Lavrač (2003); Prati et al. (2001). A parte da regra referente à *Conclusão* tem o formato Classe = $c_j$, atribuindo um valor particular $c_j$ à Classe.

Uma sintaxe alternativa de regras que é comumente utilizada é

*Classe*$\leftarrow$*Condições*,

ou na forma mais geral:

*Cabeça* $\leftarrow$ *Corpo*.

A última forma é geralmente utilizada em lógica de predicados como uma forma geral de regras na qual o *Corpo* (também chamado de antecedente) é uma conjunção de condições ou literais, e a *Cabeça* é o conseqüente que, no caso de regras if-then, é um simples literal (no caso geral, ela também pode ser uma disjunção de literais).

Um exemplo de um conjunto de regras induzidas utilizando o CN2 Clark & Niblett (1989); Clark & Boswell (1991) para o domínio de prescrição de lentes de contados é mostrado no Exemplo 3.1. Os números entre colchetes indicam a quantidade de exemplos do conjunto de treinamento, para cada uma das classes, cobertos pela regra. A classe com mais exemplos cobertos corresponde àquela prevista pela regra. A primeira e a terceira regra são consistentes, enquanto que as outras duas regras classificam erroneamente um exemplo cada uma.

Em geral, simplesmente ignoram-se os números correspondentes à distribuição de exemplos cobertos entre as classes, sendo as regras interpretadas categoricamente. Os números correspondentes à distribuição de exemplos cobertos entre as classes podem ser utilizados para medir a confiança ou significância de uma dada regra. Por exemplo, pode-se considerar a

quarta regra do Exemplo 3.1 pouco confiável pois a diferença entre o número de exemplos entre as duas classes mais cobertas (nenhuma e dura) é de apenas 1 exemplo. Mais detalhes sobre a avaliação de regras são discutidos na Seção 3.6. Além disso, esses números também poder ser utilizados como estimativa de probabilidade sobre todas as classes, ao invés de fazer apenas uma predição categórica. Por exemplo, se a descrição do paciente satisfaz a condição da segunda regra, pode-se utilizá-la para predizer uma lente macia com probabilidade $5/6 = 0.8333...$ ou nenhuma lente com probabilidade $1/6 = 0.1666...$

**Exemplo 3.1** Regras de classificação induzidas a partir do conjunto de dados lentes de contato

```
IF   Produção Lacrimal = reduzido
THEN Lentes = nenhuma         [#macia=0, #dura=0, #nenhuma=12]

IF   Produção Lacrimal = normal
 AND Astigmatismo = não
THEN Lentes = macia           [#macia=5, #dura=0, #nenhuma=1]

IF   Produção Lacrimal = normal
 AND Astigmatismo = sim
 AND Spectropia = miopia
THEN Lentes = dura            [#macia=0, #dura=3, #nenhuma=0]

IF   Produção Lacrimal = normal
 AND Astigmatismo = sim
 AND Spectropia = hipermetropia
THEN Lentes = nenhuma         [#macia=0, #dura=1, #nenhuma=2]
```

É importante ressaltar que, na segunda regra, a condição `Produção lacrimal = normal` é a negação da condição da primeira regra, e que essa restrição é incluída em todas as outras regras subseqüentes. Da mesma maneira, a condição `Astigmatismo = sim` presente na terceira regra é a negação da segunda condição na segunda regra, e assim em diante. Neste sentido, o conjunto de regras pode ser equivalentemente representado por uma lista de decisão, como mostrado no Exemplo 3.2. Alternativamente, toda a hipótese pode ser vista como uma árvore binária na qual cada ramo da esquerda leva a uma folha[2].

Conjuntos de regras, listas e árvores de decisão são formas de representação fortemente correlacionadas e todas elas são utilizadas em AM simbólico. Em geral, essas três formas de representação compartilham muitas semelhanças. Entretanto, existe uma diferença fundamental entre listas e árvores de decisão e conjuntos de regras não ordenados. Listas e árvores

[2]Note que, no caso geral, isso não necessariamente ocorre

**Exemplo 3.2** Lista de decisão induzida pelo conjunto de dados lentes de contato

```
IF Produção lacrimal = reduzido THEN Lentes = nenhuma
   ELSE /* Produção lacrimal = normal*/
        IF Astigmatismo = não THEN Lentes = macia
           ELSE /* Astigmatismo = sim*/
                IF Spectropia = miopia THEN Lentes = dura
                   ELSE /* Spectropia = hipermetropía/
                        Lentes = nenhuma
```

de decisão dividem o conjunto de exemplos em regiões disjuntas. Esse fator implica que cada exemplo é classificado por somente uma única regra (no caso da lista de decisão, a primeira regra disparada) ou ramo da árvore. Essa disjunção não ocorre, necessariamente, em conjuntos de regras não ordenados. Para regras não ordenadas, no caso de mais de uma regra disparar para um dado exemplo, as suas previsões precisam ser combinadas, por exemplo, utilizando algum esquema de votação. Além disso, assim como uma lista de decisão pode ser representada como uma árvore binária, uma árvore de decisão também pode ser representada como um conjunto de regras. Entretanto, regras não ordenadas não podem ser representadas como árvores, apenas como grafos.

## 3.2. Aprendizagem de regras como um problema de busca

Como dito na seção 3.1.2, regras geralmente assumem o formato

**if** *Condições* **then** *Conclusão*,

no qual *Condições* representa um conjunção de restrições sobre os atributos e *Conclusão* tem a forma Classe = $c_j$. Representações alternativas para uma regra são *Classe* ← *Condições*, ou *Cabeça* ← *Corpo*. Um algoritmo de aprendizagem produz uma hipótese ou modelo representado como um conjunto de regras. A construção dessa hipótese geralmente envolve quatro estágios:

**Construção da hipótese.** Para construir uma hipótese, o sistema de aprendizagem deve encontrar um conjunto de regras. Em aprendizagem envolvendo linguagens de representação de hipóteses equivalentes a lógica proposicional, esse estágio pode ser simplificado pela indução de regras seqüencialmente e independentemente, por exemplo, aplicando um algoritmo de cobertura. Em aprendizagem envolvendo linguagens de representação equivalente à lógica de primeira ordem a situação

passa a ser um pouco mais complexa no caso da recursão ser empregada, pois as regras não podem ser induzidas independentemente.

**Construção da regra.** Uma regra individual no formato *Cabeça* $\leftarrow$ *Corpo* é geralmente construída fixando-se a cabeça para um dado valor de Classe = $c_j$ e, a partir daí, heuristicamente buscar pelo melhor corpo da regra.

**Construção do corpo da regra.** Tipicamente, o corpo da regra é uma conjunção de condições. Para regras com o poder de representação equivalente à lógica proposicional, a construção do corpo da regra é geralmente feita pelo seu refinamento, adicionando condições ao corpo inicialmente vazio.

**Construção de cada condição da regras** No caso mais simples, como mencionado anteriormente, condições têm o formato de simples literais $A_i = v_{ij}$, no qual $A_i$ é um atributo e $v_{ij}$ é um dos possíveis valores que esse atributo pode assumir. No caso de atributos numéricos ou ordinais, pode-se também considerar desigualdades da forma $A_i < v$ ou $A_i > v$, no qual $v$ é um limiar a ser construído (isto é, ele pode assumir um valor não diretamente observado nos dados de treinamento). Entretanto, outras formas de construção de condição podem ser encontradas na literatura.

Surpreendentemente, a tarefa crucial de construção das condições de cada uma das regras não é muito explorada na literatura relativa à indução de regras. Entretanto, existem trabalhos na literatura a respeito da construção de atributos. A construção de condições pode ser vista como um caso particular da construção de atributos (imagine que cada condição é um atributo binário, com valor verdadeiro se esse atributo satisfaz a condição e falso caso contrário).

Induzir um conjunto de regras de classificação pode ser entendido como um problema de busca, no qual o espaço de possíveis hipóteses é determinado pela linguagem de descrição de hipóteses utilizada. Em aprendizagem de regras do tipo if-then, o espaço de hipóteses é limitado por todas as possíveis regras no formato *Classe* $\leftarrow$ *Condições*, com a *Classe* assumindo um dos seus possíveis valores, e *Condições* sendo uma conjunção de restrições para alguns atributos, como descrito anteriormente. Por simplicidade, nesta seção, nos restringiremos a condições do tipo $A_i = v_{ij}$ (a igualdade pode também ser substituída por desigualdades quando o atributo $A_i$ é contínuo).

Considerando o aprendizado de regras como um problema de busca, um critério de avaliação (por exemplo a taxa de acerto ou significância —

Seção 3.6) precisa ser definida para se decidir quando uma regra candidata (ou um conjunto de regras) é a solução encontrada para um dado problema. Enumerar todo o espaço de possíveis regras é claramente ineficiente e, dessa maneira, é necessário uma estrutura que permita explorar somente parte do espaço de hipóteses. A maioria dos algoritmos de aprendizagem indutiva simbólica estrutura o espaço pela utilização de uma noção dual de generalização e especialização (Mitchell, 1997b).

Uma das maneiras de definir generalidade é em termos de cobertura. Seja a função cobertos(R) definida como o conjunto de exemplos cobertos pela regra R. Pode-se definir a regra R como sendo mais geral que a regra R′ se:

i. ambas têm o mesmo conseqüente e,

ii. cobertos(R) $\supseteq$ cobertos(R′).

Essa definição é normalmente chamada de noção semântica da generalidade, uma vez que ela requer que sejam avaliadas regras com relação a um dado conjunto do dados. Entretanto, para o aprendizado regras do tipo if-then atributo-valor, pode-se utilizar uma simples noção sintática: dada um mesmo conseqüente, uma regra R é mais geral que uma regra R′ se o antecedente de R′ impõe, pelo menos, as mesmas restrições que R. Também se diz que R′ é mais específica que R. Para exemplificar esse conceito, considere duas regras induzidas a partir do conjunto de exemplos lentes de contato — Exemplo 3.1, na página 33.

```
IF   Produção Lacrimal = normal
 AND Astigmatismo = sim
THEN Lentes = dura           [#macia=0, #dura=4, #nenhuma=2]

IF   Produção Lacrimal = normal
 AND Astigmatismo = sim
 AND Spectropia = miopía
THEN Lentes = dura           [#macia=0, #dura=3, #nenhuma=0]
```

Claramente, a segunda regra impõe pelo menos as mesmas restrições que a primeira e é, dessa maneira, mais específica. Em termos de cobertura, enquanto a primeira regra cobre 6 exemplos da Tabela 3.2, a segunda cobre apenas 3. No caso de atributos contínuos, condições envolvendo desigualdades são comparadas da maneira óbvia: por exemplo, a condição Idade < 30 é mais geral que a condição Idade < 20, que por sua vez é mais geral que a condição Idade < 10.

Note que a generalidade ignora o rótulo da classe de cada exemplo. Em contrapartida, uma vez que uma regra mais específica cobrirá o mesmo

conjunto ou um subconjunto dos exemplos de cada classe cobertos por uma regra mais geral, tornar uma regra mais específica (ou *especializá-la*) é uma maneira de obter uma regra mais consistente que cubra exemplos somente de uma classe. Por exemplo, em comparação com a primeira regra acima, a segunda regra é capaz de remover os dois exemplos incorretamente cobertos pela primeira, mas pagando o preço de cobrir menos um exemplo corretamente coberto do que a primeira regra. O aprendizado de regras pode ser visto como um meio termo entre precisão (a proporção de exemplos corretamente cobertos) e a cobertura (a proporção de exemplos cobertos).

## 3.3. Aprendizado de regras individuais

No aprendizado de uma única regra, a maiorias dos algoritmos de aprendizado utiliza umas das seguintes estratégias:

**Geral-para-específico** ou abordagem *top-down*. Nessa categoria, os algoritmos iniciam pela regra mais geral e repetidamente especializam essa regra enquanto ela cobrir exemplos negativos. A especialização termina quando a regra não é mais inconsistente, *i.e.*, não cobre exemplos negativos. Durante a busca, algoritmos geral-para-específico asseguram que as regras induzidas cobrem pelo menos um exemplo positivo. Enquanto constroem a regra, os algoritmos utilizam um operador de refinamento que computa uma série de especializações sobre a regra.

**Específico-para-geral** ou abordagem *botom-up*. Os algoritmos que estão nessa categoria começam pela regra mais específica que cobre um dado exemplo positivo; elas então generalizam a regra até que ela não possa mais ser generalizada sem cobrir exemplos negativos.

A primeira abordagem geralmente induz regras mais gerais do que aquelas induzidas pela segunda abordagem. A busca geral-para-específico é apropriada para o aprendizado na presença de ruído porque ela pode ser facilmente guiada por heurísticas. A busca específico-para-geral é geralmente aplicável em situações nas quais poucos exemplos estão disponíveis ou para aplicações de aprendizado incremental (Flach & Lavrač, 2003).

Na abordagem geral-para-específico, o conceito de operador de refinamento é de fundamental importância. Em aprendizado a partir de exemplos no formato atributo-valor, para uma regra $R$ do tipo **if** *Condições* **then** *Conclusão*, um refinamento $R'$ de **R** é obtido pela adição de uma condição *Nova Condição* à conjunção de condições no corpo da regra, de tal maneira que a regra $R'$ torne-se **if** *Condições* **and** *Nova Condição* **then** *Conclusão*.

Utilizando o operador de refinamento, é fácil definir um procedimento de busca geral-para-específico para a indução de regras gerais. O algoritmo é descrito brevemente no Algoritmo 3.1. Esse algoritmo utiliza uma heurística para selecionar o melhor refinamento da regra atual a cada iteração. Algumas heurísticas comumente usadas incluem precisão de Laplace, entropia ou cobertura, discutidas na Seção 3.6. Essa estratégia é uma clássica busca em feixe do tipo *hill-climbing*.

**Data**: $E_i = E_i^+ \cup E_i^-$: um conjunto de exemplos positivos e negativos para uma classe $c_i$
**Result**: $Regra$
$Regra$ := Classe = $c_i \leftarrow$ *Condições*, na qual *Condições* = $\oslash$;
**repeat**
  construa um conjunto $\{R'_1, R'_2, \cdots, R'_k\}$ de $k$ refinamentos de $Regra$, na qual cada $R'_j$ é da forma $Classe = c_i \leftarrow$ *Condições* **and** *Nova Condição* ;
  avalie todas as regras de $\{R'_1, R'_2, \cdots, R'_k\}$ de acordo com um critério de qualidade;
  $Regra$ := a melhor regra $R'_j$ de $\{R'_1, R'_2, \cdots, R'_k\}$;
**until** $Regra$ *satisfaz um critério de qualidade ou não cobre nenhum exemplo negativo* ;

**Algoritmo 3.1**: Procedimento APRENDAUMAREGRA

## 3.4. Aprendizado de um conjunto de regras

Algoritmos que utilizam ambas as estratégias geral-para-específico e específico-para-geral repetem o procedimento de induzir uma única regra em um conjunto reduzido de exemplos no caso da regra construída não cobrir todos os exemplos positivos. Dessa maneira, esses algoritmos utilizam um processo iterativo para construir uma hipótese disjuntiva consistindo de um conjunto regras. Um das abordagens mais utilizadas para se construir esse conjunto é o algoritmo conhecido como cobertura de conjunto, juntamente com busca geral-para-específico, mostrado no Algoritmo 3.2. O algoritmo de cobertura de conjunto constrói repetidamente uma nova regra até que todos os exemplos positivos $E_i^+$ da classe $c_i$ sejam cobertos pelo conjunto de regras, ou algum outro critério de qualidade pré-definido é satisfeito. Uma vez que uma regra é adicionada ao modelo, todos os exemplos positivos cobertos por aquela regra são removidos do conjunto de exemplos positivos. Para encontrar a melhore regra, o procedimento APRENDAUMAREGRA é aplicado.

**Data**: $E_i = E_i^+ \cup E_i^-$: um conjunto de exemplos positivos e negativos para uma classe $c_i$
**Result**: $ConjuntoDeRegras$
$ConjuntoDeRegras := \oslash$;
**repeat**
  $Regra$ := APRENDAUMAREGRA($E_i$);
  $ConjuntoDeRegras := ConjuntoDeRegras \cup Regra$;
  $\widehat{E_i^+}$ := conjunto de exemplos positivos cobertos por $Regra$ ;
  $E_i^+ := E_i^+ \setminus \widehat{E_i^+}$ ;
**until** $ConjuntoDeRegras$ *satisfaz um critério de qualidade ou* $E_i^+$ *está vazio* ;

**Algoritmo 3.2**: Procedimento APRENDACONJUNTODEREGRAS

O algoritmo de cobertura de conjunto pode ser entendido da seguinte maneira: cada regra no conjunto de regras cobre uma região do espaço de exemplos e atribui a essa região uma classe. O papel de um exemplo positivo (para uma dada classe) é forçar a atenção do algoritmo de aprendizado para uma área particular do espaço de exemplos. Já o papel dos exemplos negativos é previnir uma super-generalização, *i.e.*, assegurar que outras áreas do espaço de exemplos não sejam cobertas pela regra. Exemplos positivos devem ser cobertos por pelo menos uma regra, enquanto que exemplos negativos não deveriam ser cobertos por nenhuma regra. Esse fato explica porque quando se constrói uma nova regra somente os exemplos positivos são removidos. Os exemplos negativos são deixados para previnir super-generalização na construção de novas regras.

Deve ser notado que a remoção de exemplos durante o treinamento distorce as estatísticas do conjunto de aprendizado e introduz uma certa dependência de ordem entre as regras. Por exemplo, a última regra aprendida é muito dependente das regras previamente aprendidas e os exemplos positivos que elas cobrem. Essa última regra pode não ser estatisticamente significante, pois foi induzida com poucos exemplos. Uma variação do algoritmo de cobertura de conjunto pode ser obtida pela introdução de pesos aos exemplos (Cohen & Singer, 1999), os quais decrescem a medida que o número de regras cresce.

### 3.4.1 Conjunto de regras não ordenadas

O procedimento descrito anteriormente é apropriado para a resolução de problemas com classes binárias. Ele induz regras somente para a classe positiva, e se uma delas dispara, o exemplo é classificado como positivo. Se nenhuma das regras disparar, ele é classificado como negativo. Para problemas multiclasse, para se induzir regras para todas as classes $c_i$, $i = 1..n_{cl}$ o procedimento APRENDACONJUNTODEREGRAS é repetido para cada uma

das classes. A cada iteração uma das classes é designada como positiva e o restante das classes são agrupadas na classe negativa. Dessa maneira, são criados vários conjuntos de regras para cada uma das classes. Esses conjuntos de regras são então reunidos em um único conjunto para formar o conjunto final de regras. Esse procedimento é conhecido como *indução de regras não ordenadas*.

Quando da classificação de um novo exemplo, todas as regras são testadas e a predição de todas as regras que cobrem o exemplo são coletadas. Um mecanismo de votação é geralmente utilizado para decidir qual é a classificação final do exemplo. Por exemplo, no sistema CN2 (Clark & Boswell, 1991), as classificações conflitantes são resolvidas levando-se em consideração o número de exemplos de cada classe utilizados para induzir o modelo coberto por cada uma das regras. Por exemplo, considere que tenhamos um problema de três classes e as duas regras que cobrem um determinado exemplo que se queira classificar têm a seguinte cobertura de exemplos no conjunto de treinamento: [10, 1, 2] e [4, 15, 0], *i.e.*, a primeira regra cobre 10 exemplos da primeira classe, um da segunda e dois da terceira e a segunda regra cobre 4 exemplos da primeira classe, 15 da segunda e nenhum da terceira. A cobertura "somada" seria [14, 16, 2] e o exemplo seria classificado como sendo da segunda classe, pois ela é a mais freqüente na cobertura "somada".

### 3.4.2 Conjuntos de regras ordenadas

Uma outra abordagem comum para alguns algoritmos de aprendizado de regras é a indução de regras não ordenadas ou listas de decisão. Quanto ao algoritmo que induz a lista de decisão, a maior diferença com relação ao Algoritmo 3.2 está no fato de que ao invés de remover apenas os exemplos corretamente cobertos pela regra, todos os exemplos cobertos por ela são removidos. A razão está relacionada ao fato que a classificação de um novo exemplo em uma lista de decisão é dada pela primeira regra que cobre àquele exemplo. Dessa maneira, não é possível que novas regras cubram os exemplos incorretamente cobertos pela regra, e não há a necessidade de deixá-los no conjunto de treinamento.

## 3.5. Aprendizado de árvores de decisão

Árvore de Decisão (AD) é um dos métodos mais consagrados em aprendizado de máquina simbólico supervisionado. Algoritmos que induzem árvores de decisão pertencem à família de algoritmos *Top Down Induction of*

*Decision Trees* (TDIDT). Na Figura 3.1 é mostrado um exemplo de uma árvore de decisão induzida a partir do conjunto de dados na Tabela 3.2. Uma árvore de decisão é uma estrutura de dados definida recursivamente como:

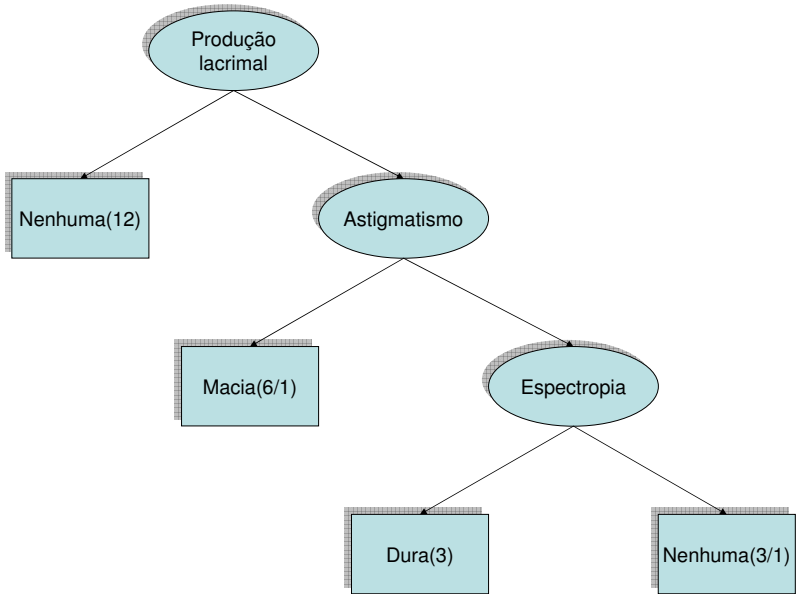



**Figura 3.1:** Exemplo de árvore de decisão

- um *nó folha* que corresponde a uma classe ou
- um *nó de decisão* que contém um teste sobre algum atributo (condição). Para cada resultado do teste existe uma aresta para uma subárvore. Cada subárvore tem a mesma estrutura que a árvore.

Para classificar um novo exemplo, basta começar pela raiz da árvore (nó inicial da árvore), seguindo cada nó de decisão de acordo com o valor do atributo do novo exemplo até que uma folha seja alcançada. Quando uma folha é alcançada, a classificação é dada pela classe correspondente ao nó folha.

O método para a construção de uma árvore de decisão a partir de um conjunto de treinamento $E$ é surpreendemente simples. Assumindo que as classes do conjunto de exemplos de treinamento são $\{c_1, c_2, \ldots, c_{n_{cl}}\}$, os seguintes passos devem ser seguidos:

1. *$E$ contém um ou mais exemplos, todos pertencentes à mesma classe $c_j$.* Nesse caso, a árvore de decisão para $E$ é um nó folha rotulado pela classe $c_j$;
2. *$E$ não contém exemplos.* Novamente, nessa situação, a árvore é uma folha mas a classe associada à folha deve ser determinada a partir de

informação além de $E$. Por exemplo, a classe mais freqüente para o nó-pai desse nó pode ser utilizada;

3. *$E$ contém exemplos que pertencem a várias classes.* Nesse caso, a idéia é dividir $E$ em subconjuntos de exemplos que são (ou aparentam ser) conjuntos de exemplos mais "puros", *i.e.*, conjuntos com um número maior de exemplos de uma única classe. Normalmente, um teste é escolhido baseado em um único atributo que possui resultados mutuamente exclusivos (na realidade, cada sistema tem sua própria forma de escolher o atributo que será utilizado no teste). Sejam os possíveis resultados do teste denotados por $\{O_1, O_2, ..., O_r\}$. $E$ é então particionado em subconjuntos $E_1$, $E_2$, ..., $E_r$, nos quais cada $E_i$ contém todos os exemplos em $E$ que possuem como resultado daquele teste o valor $O_i$. A árvore de decisão para $E$ consiste em um nó interno identificado pelo teste escolhido e uma aresta para cada um dos resultados possíveis;

4. Os passos 1, 2 e 3 são aplicados recursivamente para cada subconjunto de exemplos de treinamento de maneira que, em cada nó, as arestas levam para as subárvores construídas a partir do subconjunto de exemplos $E_i \subseteq E$.

O ponto principal de um algoritmo de aprendizado por AD, *i.e.*, algoritmos que usam AD como linguagem de descrição de hipóteses, depende do critério utilizado para escolher o atributo que particiona o conjunto de exemplos em cada iteração. Algumas possibilidades para escolher esse atributo são:

**aleatória** seleciona qualquer atributo aleatoriamente;

**menos valores** seleciona o atributo com a menor quantidade de valores possíveis;

**mais valores** seleciona o atributo com a maior quantidade de valores possíveis;

**ganho máximo** seleciona o atributo que possui o maior ganho de informação esperado, *i.e.*, seleciona o atributo que resultará no menor tamanho esperado das subárvores, assumindo que a raiz é o nó atual;

**índice Gini** seleciona o atributo baseado na estatística Gini, utilizado no sistema CART (Breiman et al., 1984) e

**razão de ganho** seleciona o atributo ponderando o ganho de informação esperado em relação ao nó pai, utilizado no sistema C4.5 (Quinlan, 1993).

Após a construção da árvore de decisão, é possível que a árvore induzida seja muito específica para o conjunto de treinamento. Nesse caso, diz-se que a AD superajustou os dados de treinamento, ou seja, ocorreu um *overfitting*. Como os exemplos de treinamento são apenas uma amostra de todos os possíveis exemplos, é possível adicionar na árvore arestas que melhoram seu desempenho nos dados de treinamento, mas que piora seu desempenho em um conjunto de teste. Para tentar solucionar o problema de superajuste dos dados, alguns sistemas *podam* a AD depois de induzí-la. Esse processo reduz o número de nós internos, reduzindo a complexidade da árvore e, possivelmente, melhorando o desempenho da árvore original. Esse tipo de poda é chamado de *pós-poda* uma vez que ele ocorre após a indução da AD. Existem vários métodos de pós-poda, incluindo complexidade do erro (Breiman et al., 1984) e erro pessimista (Quinlan, 1993). É também possível utilizar *pré-poda* na AD. Esse processo é efetuado enquanto a AD é induzida. Entretanto, a pré-poda sofre um efeito colateral: conjunções de teste podem ser a melhor forma de particionar os exemplos, mas seus atributos individuais podem não distinguir muito bem os exemplos. Assim, a pré-poda pode evitar que determinados tipos de conjunções apareçam na árvore.

Uma AD pode ser facilmente mapeada em um conjunto de regras, transformando cada ramo da árvore (cada caminho da raiz até cada um dos nós-folha) em uma regra. As regras transcritas de uma árvore de decisão são disjuntas, isto é, apenas uma única regra dispara quando um novo exemplo é classificado.

## 3.6. Avaliando a qualidade de regras

Várias medidas podem ser usadas para avaliar o desempenho de um modelo de classificação, sendo a precisão a mais comum. Entretanto, em problemas do mundo real, nem sempre é possível encontrar um modelo que tenha uma boa precisão para classificar novos exemplos. Quando as regras são analisadas individualmente, freqüentemente algumas dessas regras cobrem muita bem uma parte do espaço de exemplos. Nesse caso, além de medir a precisão do modelo como um todo, é possível avaliar separadamente cada uma das regras que constituem o modelo. Nesse contexto, regras podem ser avaliadas com o objetivo de saber quais são aquelas melhor sustentadas pelos dados ou, ainda, podem ser avaliadas com o intuito de selecionar aquelas que possam trazer algum conhecimento surpreendente ou inesperado.

A maioria das medidas de avaliação de regras estão baseadas na matriz de contingência para cada regra (Freitas, 1999; Prati et al., 2002). Considerando cada regra no formato *Corpo* $\rightarrow$ *Cabeça*, ou resumidamente $B \rightarrow H$[3], sua correspondente matriz de confusão é mostrada na Tabela 3.3 (Lavrač et al., 1999). Nessa tabela, $B$ denota o conjunto de exemplos para os quais a condição da regra é verdadeira e seu complemento $\bar{B}$ denota o conjunto de exemplos para os quais a condição da regra é falsa; analogamente para $H$ e $\bar{H}$. $BH$ denota o conjunto de exemplos $B \cap H$ no qual ambos $B$ e $H$ são verdadeiros, $B\bar{H}$ representa o conjunto de exemplos $B \cap \bar{H}$ no qual $B$ é verdadeiro e $H$ é falso e assim por diante. Por generalidade, denota-se a cardinalidade de um conjunto $A$ por $a$, ou seja, $a = |A|$. Assim, $b$ denota o número de exemplos no conjunto $B$, ou seja, $b = |B|$, $h$ denota o número de exemplos no conjunto $H$, ou seja $h = |H|$, $bh$ denota o número de exemplos no conjunto $BH$, ou seja, $bh = |BH|$ e assim por diante, e $n_{ex}$ indica o número total de exemplos.

Matriz de contingência

| | $H$ | $\overline{H}$ | |
|---|---|---|---|
| $B$ | $hb$ | $\overline{h}b$ | $b$ |
| $\overline{B}$ | $h\overline{b}$ | $\overline{hb}$ | $\overline{b}$ |
| | $h$ | $\overline{h}$ | $n_{ex}$ |

$hb$ = número de exemplos para os quais $H$ é verdade e $B$ é verdade
$\overline{h}b$ = número de exemplos para os quais $H$ é falso e $B$ é verdade
$h\overline{b}$ = número de exemplos para os quais $H$ é verdade e $B$ é falso
$\overline{hb}$ = número de exemplos para os quais $H$ é falso e $B$ é falso
$b$ = número de exemplos para os quais $B$ é verdade
$\overline{b}$ = número de exemplos para os quais $B$ é falso
$h$ = número de exemplos para os quais $H$ é verdade
$\overline{h}$ = número de exemplos para os quais $H$ é falso
$n_{ex}$ = número total de exemplos

**Tabela 3.3:** Matriz de contingência para uma regra

Denotando-se por $p(A)$ a freqüência relativa $frac|A|n_{ex} = \frac{a}{n_{ex}}$ associada ao conjunto $A$, no qual $A$ é um subconjunto dos $n_{ex}$ exemplos, podemos utilizar essa freqüência relativa como uma estimativa de probabilidade. A notação $p(A|B)$ segue sua definição habitual de probabilidade condicional em teoria da probabilidade, dada pela Equação (3.1), onde $A$ e $B$ são ambos subconjuntos do conjunto de $n_{ex}$ exemplos.

$$P(A|B) \simeq p(A|B) = \frac{p(A \cap B)}{p(B)} = \frac{p(AB)}{p(B)} = \frac{\frac{|AB|}{n}}{\frac{|B|}{n}} = \frac{\frac{ab}{n}}{\frac{b}{n}} = \frac{ab}{b} \qquad (3.1)$$

[3] $Body \rightarrow Head$

### 3.6.1 Medidas simples

Utilizando como base a matriz de contingência, é possível definir a maioria das medidas sobre regras, como, a precisão ($Acc$), erro ($Err$), confiança negative ($NegRel$), sensitividade ($Sens$), especificidade ($Spec$), cobertura ($Cov$) e suporte ($Sup$), definidas na Tabela 3.4.

| | |
|---|---|
| Precisão | $Acc(B \rightarrow H) = P(H\|B) = \frac{hb}{b}$ |
| Erro | $Err(B \rightarrow H) = P(\overline{H}\|B) = \frac{\overline{h}b}{b}$ |
| Confiança negativa | $NegRel(B \rightarrow H) = P(\overline{H}\|\overline{B}) = \frac{\overline{hb}}{\overline{b}}$ |
| Sensitividade | $Sens(B \rightarrow H) = P(B\|H) = \frac{hb}{h}$ |
| Especificidade | $Spec(B \rightarrow H) = P(\overline{B}\|\overline{H}) = \frac{\overline{hb}}{\overline{h}}$ |
| Cobertura | $Cov(B \rightarrow H) = P(B) = \frac{b}{n}$ |
| Suporte | $Sup(B \rightarrow H) = P(HB) = \frac{hb}{n}$ |

**Tabela 3.4:** Medidas simples de avaliação de regras

A *precisão* de uma regra, também chamada de confiança, é uma medida do quanto essa regra é específica para o problema. O *erro* de uma regra é o complemento da precisão. A *confiança negativa* de uma regra é o correspondente à precisão, mas para os exemplos que não são cobertos pela regra. A *sensitividade* de uma regra é semelhante ao *recall* de casos positivos usados em recuperação de informação; também conhecida como *completeza*, é uma medida do número (relativo) de exemplos da classe prevista em $H$ cobertos pela regra. A *especificidade* de uma regra é o correspondente à completeza, mas para os exemplos que não são cobertos pela regra. A *cobertura* de uma regra é uma medida do número (relativo) de exemplos cobertos pela regra. O *suporte* de uma regra é uma medida do número (relativo) de exemplos cobertos corretamente pela regra.

### 3.6.2 Precisão *versus* generalidade

Cada uma das medidas apresentadas na seção anterior tem como objetivo avaliar um aspecto de uma regra. A precisão, por exemplo, tem como objetivo minimizar o número de exemplos incorretamente cobertos pela regra. Entretanto, isso pode levar a casos patológicos, como uma regra muito precisa que cobre apenas um único exemplo. Além disso, dadas duas regras com a mesma precisão, a regra mais geral das duas, ou seja, a que cobre mais exemplos, é a mais preferível. Nenhuma das medidas apresentadas na seção anterior são capazes de fazer essa distinção.

Uma das maneiras de gerenciar o compromisso entre precisão e generalidade é derivar medias compostas. Geralmente, escolhe-se medidas "ortogonais", tais como a precisão e sensitividade, que dão origem à medida F (van Rijsbergen, 1979), definida pela Equação 3.2. Essa medida tem um parâmetro $\alpha$ que indica a importância relativa de cada uma das duas medidas.

$$F_\alpha(B \rightarrow H) = \frac{(\alpha + 1) \times Acc \times Sens}{Sens + \alpha \times ACC} \tag{3.2}$$

Uma outra medida comumente utilizada para ponderar precisão e cobertura é a cobertura relativa com pesos (Lavrač et al., 2004), também conhecida como novidade. Essa medida é apresentada na Equação 3.3.

$$WRacc(B \rightarrow H) = Nov(B \rightarrow H) = p(B)(p(H|B) - P(H)) = p(HB) - P(H)P(B) \tag{3.3}$$

Essa medida tem a propriedade de preferir regras um pouco menos precisas, mas com uma maior cobertura do que uma regra muito precisa mas que cobre somente alguns exemplos

Alternativamente, regras podem ser representadas em gráficos do tipo precisão-sensitividade (*precision-recall* como são mais conhecidos) ou gráficos ROC, que permitem comparar regras e determinar as condições em que uma determinada regra é melhor ou pior que outra. Análise ROC é apresentada no Capítulo 4.

### 3.6.3 Estimando probabilidades

As regras são geralmente avaliadas em um conjunto de treinamento. No entanto, estamos geralmente interessados em estimar o seu desempenho em toda a população. Anteriormente foi assumido por simplicidade que as estimativas eram realizadas utilizando-se freqüências. Essa abordagem pode, no entanto, gerar estimativas pouco confiáveis. Dois métodos são geralmente descritos na literatura como alternativas para estimar probabilidades: a correção de Laplace e a correção m. Ambas aplicam uma correção à freqüência relativa, que pode ser interpretada como se houvesse uma distribuição de exemplos *a priori* sobre a variável sendo estimada. Na correção de Laplace, essa distribuição é uniforme e na correção m é uma generalização que é capaz de levar em consideração uma certa probabilidade *a priori* a respeito das classes.

Por simplicidade, a seguir nos restringimos à discussão da precisão de uma regra, mas o procedimento é aplicado a casos arbitrários no qual a probabilidade deve ser estimada a partir de freqüências. Suponha que uma regra cubra $pos$ exemplos positivos e $neg$ exemplos negativos. A estimativa

da precisão com freqüências pode ser computada como $\frac{pos}{pos+neg}$. Por exemplo, se a regra cobre apenas um exemplo positivo corretamente, essa estimativa não é muito robusta pois qualquer alteração, mesmo que pequena, digamos um exemplo positivo ou negativo coberto pela regra, provocará uma grande mudança na estimativa.

A correção de Laplace é obtida pela inserção de um exemplo "fictício" ou "virtual" para cada classe. Dessa maneira, a precisão estimada a partir da correção de Laplace torna-se $\frac{pos+1}{pos+neg+k}$, na qual $k$ é o número de classes. Por exemplo, em um problema de duas classes, a precisão para $pos = 1$, $neg = 0$ estimada a partir das freqüências é $100,00\%$. Com a correção de Laplace, a precisão estimada é de $66,66\%$. Imagine que a cobertura de exemplos negativos dessa regra seja 1 e não zero, *i.e.*, $neg = 1$. A estimativa por freqüências, que era de $100\%$, cai drasticamente para $50\%$. Já a correção de Laplace, também de $50\%$, oscila menos (de $66.66\%$ para $50\%$). A medida que a cobertura da regra aumenta, tanto a estimativa por freqüência quanto a corrigida tendem a valores semelhantes.

A correção m generaliza a correção de Laplace assumindo uma probabilidade *a priori* $pos_i$ para cada classe $c_i$. A correção m é igual a $\frac{pos+m\times pos_i}{pos+neg+m}$, no qual $m$ é um parâmetro. Isso é equivalente a adicionar $m$ exemplos virtuais distribuídos de acordo com a probabilidade *a priori* das classes. Observe que a correção de Laplace é um caso especial da correção m, no qual $m = k$ e $pos_i = 1/k$ para todas as classes. O parâmetro $m$ controla o papel das distribuições de probabilidade e das evidências provenientes dos exemplos: altos valores de $m$ dão um maior peso às probabilidades à priori e menos aos exemplos. Altos valores de $m$ são aconselháveis para conjuntos de dados com muito ruído. A probabilidade *a priori* $pos_i$ pode ser estimada a partir do conjunto de treinamento utilizando-se as freqüências relativas de cada classe.

## 3.7. Considerações finais

Neste capítulo foi feita uma introdução ao aprendizado de regras proposicionais, utilizado no desenvolvimento deste trabalho. Foram abordadas as duas principais maneiras de se induzir regras: indução de regras diretamente e indução de árvores de decisão. A indução de regras é também conhecida como abordagem separar-para-conquistar pois, uma vez aprendida uma regra, os exemplos por ela cobertos são removidos. A indução de árvores de decisão é também conhecida como estratégia dividir-para-conquistar, pois o conjunto de exemplos é recursivamente dividido à medida que a árvore é induzida.

Aprendizado de regras tem uma longa história dentro da área de aprendizado de máquina. Representantes da família separar-para-conquistar aparecem na literatura desde que aprendizado de máquina se firmava como área de pesquisa (Michalski, 1969). A principal razão para o interesse em aprendizado de regras está relacionado à atratividade de regras como um formalismo de representação de conceitos facilmente compreensíveis.

Capítulo 4

# Análise ROC

VÁRIOS dos trabalhos desenvolvidos durante o doutoramento estão relacionados à análise ROC, seja na avaliação ou na construção de modelos. Neste capítulo é apresentada uma breve introdução à análise ROC. Este capítulo está organizado da seguinte maneira: na Seção 4.1 são apresentadas algumas considerações iniciais sobre a análise ROC. Na Seção 4.2 é apresentado o gráfico ROC. Finalmente, na Seção 4.3, são apresentadas as considerações finais deste capítulo.

## 4.1. Considerações iniciais

Análise ROC — do inglês *Receiver Operating Characteristic* — é um método gráfico para avaliação, organização e seleção de sistemas de diagnóstico e/ou predição. Gráficos ROC foram originalmente utilizados em detecção de sinais, para se avaliar a qualidade de transmissão de um sinal em um canal com ruído (Egan, 1975). Gráficos ROC também são muito utilizada em psicologia para se avaliar a capacidade de indivíduos distinguirem entre estímulo e não estímulo (Green & Swets, 1989); em medicina, para analisar a qualidade de um determinado teste clínico (Zhou et al., 2002); em economia (onde é conhecida como gráfico de Lorenz), para a avaliação

de desigualdade de renda (Gastwirth, 1971); e em previsão do tempo, para se avaliar a qualidade das predições de eventos raros (Mylne, 2002).

Recentemente, a análise ROC foi introduzida em aprendizagem de máquina e mineração de dados como uma ferramenta útil e poderosa para a avaliação de modelos de classificação (Bradley, 1997; Spackman, 1989). Ela é particularmente útil em domínios nos quais existem uma grande desproporção entre as classes ou quanto deve-se levar em consideração diferentes custos/benefícios para os diferentes erros/acertos de classificação. Análise ROC também tem sido utilizada para a construção (Prati & Flach, 2005) e refinamento de modelos (Flach & Wu, 2005).

### 4.1.1 Probabilidade conjunta e condicional

A seguir, restringiremos nossa discussão a problemas binários de classificação, *i.e.*, que tenham somente duas classes). Sem perda de generalidade, denominaremos cada uma das classes como positiva e negativa. Uma maneira natural de apresentar as estatísticas para a avaliação de um modelo de classificação é por meio de uma tabulação cruzada entre a classe prevista pelo modelo e a classe real dos exemplos. Essa tabulação é conhecida como tabela de contingência (também chamada de matriz de confusão). Na Tabela 4.1(a) é mostrada uma matriz de contingência com freqüências absolutas (contagem). Nessa tabela, $TP$, $FP$, $FN$ e $TN$ correspondem, respectivamente, as quantidade de verdadeiro/falso[1] positivo/negativo. $PP$, $PN$ correspondem ao número de exemplos preditos como positivos/negativos e $POS/NEG$ ao número de exemplos reais positivos/negativos. $N$ é o tamanho da amostra. Note que essa matriz é similar a apresentada na Tabela 3.3, só que ela refere a um modelo completo, enquanto que a da Tabela 3.3 refere-se a apenas uma regra.

(a) Freqüência absoluta

| | **predito** | | |
|---|---|---|---|
| **real** | $TP$ | $FN$ | $POS$ |
| | $FP$ | $TN$ | $NEG$ |
| | $PP$ | $PN$ | $N$ |

(b) Probabilidade conjunta

| | $Y$ | $\overline{Y}$ | |
|---|---|---|---|
| $X$ | $p(X,Y)$ | $p(X,\overline{Y})$ | $p(X)$ |
| $\overline{X}$ | $p(\overline{X},Y)$ | $p(\overline{X},\overline{Y})$ | $p(\overline{X})$ |
| | $p(Y)$ | $p(\overline{Y})$ | 1 |

**Tabela 4.1:** Matriz de contingência para modelos de classificação

Se dividirmos cada entrada na matriz mostrada na Tabela 4.1(a) pelo tamanho da amostra, cada entrada dessa matriz representa uma estimativa

[1] Quando um exemplo cuja classe real é positiva é classificado como positivo, ele é denominado verdadeiro positivo. Caso a classe real seja negativa e ele é classificado como positivo, ele é denominado falso positivo. Notação similar é empregada no caso dos exemplos negativos

da probabilidade conjunta da classe real do exemplo e da predição dada pelo modelo[2]. Essa nova matriz é mostrada na Tabela 4.1(b), na qual $X$ representa a variável aleatória **classe real do exemplo = positiva** e $Y$ representa a variável **classe predita do exemplo = positiva**. $\overline{X}$ e $\overline{Y}$ representam a negação de $X$ e $Y$.

Como a matriz mostrada na Tabela 4.1(b) é apenas uma re-escala da matriz mostrada na Tabela 4.1(a), elas são equivalentes. Além disso, toda a informação necessária para avaliar o modelo está contida nessas matrizes. No entanto, uma análise mais refinada pode ser feita pela decomposição das probabilidades conjuntas em probabilidades condicionais e marginais. Probabilidades condicionais podem ser obtidas a partir das leis básicas de probabilidade:

$$P(X,Y) = P(X|Y)P(Y) = P(Y|X)P(X),$$

na qual $P(X|Y)$, como mencionado anteriormente, é a probabilidade condicional de $X$ ser verdade, dado que $Y$ é verdade, sendo que $P(X,Y) = P(Y,X)$, mas $P(X|Y) \neq P(Y|X)$. Dado um conjunto de exemplos, todas essas probabilidades podem ser estimadas como proporções. Mais especificamente,

$$P(X|Y) \equiv \frac{p(X,Y)}{p(Y)} = \frac{TP}{PP} \text{ e } P(Y|X) \equiv \frac{p(X,Y)}{p(X)} = \frac{TP}{POS}.$$

Essas probabilidades condicionais também podem ser calculadas para as quatro entradas da matriz, e também podem ser representadas como matrizes.

$P(X|Y)$ é importante para o usuário do modelo, uma vez que ela dá a probabilidade de que a classe seja $X$, dado que a previsão feita pelo modelo é $Y$. Essa probabilidade também é conhecida como *confiança*. Entretanto, em termos de avaliação do modelo, $P(Y|X)$ é muito mais útil. Uma das vantagens da fatoração de $P(X,Y)$ em $P(Y|X)$ e $P(X)$ é que $P(Y|X)$ é condicional ao valor de $X$, *i.e*, é condicional à proporção de exemplos entre as classes. Essa probabilidade condicional é freqüentemente conhecida como *crença* ou *verossimilhança*, uma vez que ela especifica a probabilidade de que uma predição particular é feita dado a ocorrência de uma observação específica. Essa probabilidade indica o quanto um modelo é capaz de discriminar os casos entre as possíveis classes.

Além disso, as probabilidades marginais de $X$ são as únicas que não envolvem, de maneira alguma, as previsões do modelo. Por esse motivo,

[2]Para se obter uma estimativa mais confiável, em amostras grandes é recomendável a utilização de um conjunto independente de exemplos de teste. Caso o tamanho da amostra seja pequena, geralmente utiliza-se métodos de re-amostragem, tal como validação cruzada.

a distribuição de $X$ é geralmente assumida como uma característica do domínio e não dependente do modelo. Feita essa assumpção, somente dois valores são necessários para descrever a matriz de contingência, pois $P(Y|X) = 1 - P(\overline{Y}|X)$ e $P(Y|\overline{X}) = 1 - P(\overline{Y}|\overline{X})$. Essas duas probabilidades são independentes da proporção de exemplos *a priori* entre as classes. Como será abordado na Seção 4.1.2, essa é uma propriedade importante na avaliação de modelos em domínios com classes desbalanceadas e/ou diferentes custos de classificação.

### 4.1.2 Avaliação de modelos

A avaliação de um modelo de classificação é baseada na análise da matriz de contingência (ou de suas derivações). Uma das maneiras mais comuns de avaliar modelos é a derivação de medidas que, de alguma maneira, tentam medir a "qualidade" do modelo. Reduzir a matriz de contingência a uma única medida tem algumas vantagens aparentes. A principal delas é que é mais fácil escolher o "melhor" em termos de um único valor. Entretanto, é comum encontrar casos em que uma dada medida é apropriada para um problema, mas ela é irrelevante para outros. Também, é comum encontrar situações em que a avaliação é um problema de múltiplas faces, nas quais é possível definir várias medidas, sendo perfeitamente possível que um modelo seja melhor que outro para algumas dessas medidas, mas seja pior com relação a outras medidas. Nesses casos, utilizar uma única medida pode dar a falsa impressão de que o desempenho pode ser avaliado utilizando apenas essa medida.

Tomemos como exemplo a taxa de erro de classificação — uma das medidas mais comuns utilizadas em aprendizado de máquina. Existem várias situações em que a taxa de erro de classificação não é apropriada para a avaliação de modelos de classificação. Uma situação comum se dá quando o número de exemplos em cada uma das classes é muito desbalanceado (Batista et al., 2004). Por exemplo, suponha que em um dado domínio o número de exemplos de uma das classes seja 99%. Nesse caso, é comum se obter baixas taxas de erro — um modelo que sempre retorna a classe majoritária terá uma taxa de erro de apenas 1%. No entanto, esse modelo que sempre classifica um novo exemplo na classe majoritária não irá acertar nenhuma classificação de exemplos da classe minoritária.

Além disso, a taxa de erro assume custos iguais para os erros em ambas as classes. Em muitos domínios é comum haver diferentes custos de classificação para as diferentes classes. Em medicina, por exemplo, o custo de classificar incorretamente um paciente doente como sadio para uma dada doença grave é muito maior do que classificar um paciente sadio como do-

ente pois, no primeiro caso, a falha no diagnóstico pode levar à morte do paciente. Em se conhecendo os custos de classificação, esse problema pode ser remediado pela substituição da taxa de erro pelo custo médio esperado (Elkan, 2001a). Entretanto, esses custos geralmente não são conhecidos, ou até mesmo podem variar, dependendo de fatores externos.

Um outro problema é que tanto a taxa de erro quanto o custo médio esperado são dependente da distribuição de exemplos entre as classes. Na maioria das aplicações de AM e MD assume-se que os exemplos disponíveis para o treinamento representam uma amostra natural da população de casos. Por amostragem natural entende-se que a proporção de exemplos amostrados para cada uma das classes é representativa com relação à população de interesse (Fleiss, 1981). Em outras palavras, assume-se que não existe nenhum vício na amostra e que a proporção de exemplos para cada uma das classes aproxima bem a proporção de casos na população da qual os exemplos foram amostrados. Essa assumpção, mesmo que plausível, não é, necessariamente, correta. Suponha que queiramos construir um modelo para reconhecer se uma dada seqüência de aminoácidos corresponde ou não a um gene humano. Nesse caso, a amostra não pode ter a proporção real de exemplos, pois não se conhece o número total de genes que formam o genoma humano[3]. Nesse caso nem a taxa de erro nem o custo médio esperado têm o significado a eles atribuídos, pois variações na proporção de exemplos (diferente estimativas do número de genes) irão alterar os valores das medidas de desempenho, mesmo que o desempenho global do modelo não mude.

Essas deficiências não são exclusivas da taxa de erro. Qualquer medida que tenha como objetivo reduzir a avaliação de um modelo de classificação a um único valor terá, em maior ou menor grau, uma perda de informação. Geralmente, a não ser que se tenha domínios com critérios para avaliação claramente definidos e estáticos, a avaliação de um modelo utilizando uma única medida pode levar a conclusões errôneas. Em outras palavras, não existe uma única medida boa, a não ser que seja possível definir, para aquele domínio, o significado de bom.

Ainda que consideremos apenas o caso em que tanto observações quanto previsões são binárias, muitos sistemas de aprendizado fornecem um valor contínuo para as previsões, mesmo no caso em que as classes são discretas. Por exemplo, redes neurais geralmente produzem valores entre zero e um. Para se obter uma classificação, geralmente coloca-se um limiar na variável de predição. Todas as predições acima desse limiar são atribuídas a uma das classes, e as predições abaixo são atribuídas à outra. Dessa maneira,

[3]Esse número já foi estimado em 100.000, 70.000 e 50.000, entre outros. A estimativa atual é entre 30.000 e 40.000.

para um dado limiar, é possível se obter uma previsão binária pela discretização da previsão contínua. Essa discretização contribui com um outro problema para a avaliação do modelo de classificação: a escolha do limiar é arbitrária e geralmente baseada na taxa de erro. Assim, cada possível limiar produz uma matriz de contingência diferente, e, por conseqüência, diferentes valores para as medidas.

Em termos de avaliação, ao invés de estabelecer um limiar de classificação e avaliar o modelo de classificação derivado desse limiar, é mais interessante avaliar como o modelo ordena os exemplos. Um bom modelo deve ordenar os exemplos de tal forma que, observando-se a variável de predição, exemplos de classes semelhantes sejam agrupados em faixas contínuas de valores. É importante ressaltar que, como é possível derivar uma classificação colocando-se um limiar na variável contínua, avaliar como os exemplos são ordenados é mais vantajoso pois é independente do limiar e engloba a avaliação da classificação.

Também é importante ressaltar que, assim como na taxa de erro (e outras medidas), é possível derivar estatísticas a respeito da qualidade da ordenação. Uma dessas estatísticas é o teste de ordenação de Wilconxon. Como será visto na Seção 4.2, essa estatística tem uma correlação com a análise ROC. Entretanto, é importante ressaltar que, da mesma maneira que as medidas para se avaliar o modelo de classificação, avaliar a ordenação dos exemplos com uma única medida também tem as suas desvantagens.

## 4.2. O gráfico ROC

Uma alternativa à avaliação utilizando medidas é uso de gráficos e/ou diagramas. Gráficos permitem uma melhor visualização da multidimensionalidade do problema de avaliação. O gráfico ROC é baseado na probabilidade de detecção, ou taxa de verdadeiros positivos (*tpr* $= P(Y|X)$) e a probabilidade de falsos alarmes, ou taxa de falsos positivos (*fpr* $= P(Y|\overline{X})$). Para se construir o gráfico ROC plota-se *fpr* no eixo dos ordenadas — eixo $x$ — e *tpr* no eixo das abscissas — eixo $y$.

Um modelo de classificação é representado por um ponto no espaço ROC. Alguns pontos no espaço ROC merecem destaque. O ponto $(0, 0)$ representa a estratégia de nunca classificar um exemplo como positivo. Modelos que correspondem a esse ponto não apresentam nenhum falso positivo, mas também não conseguem classificar nenhum verdadeiro positivo. A estratégia inversa, de sempre classificar como positivo, é representada pelo ponto $(100\%, 100\%)$. O ponto $(100\%, 0)$ representa o modelo perfeito, *i.e.*, todos os

exemplos positivos e negativos são corretamente classificados. O ponto $(100\%, 0)$ representa o modelo que sempre faz predições erradas. Modelos próximos ao canto inferior esquerdo podem ser considerados "conservativos": eles fazem uma classificação positiva somente se têm grande segurança na classificação. Como conseqüência, eles cometem poucos erros falsos positivos, mais freqüentemente têm baixas taxas de verdadeiros positivos. Modelos próximos ao canto superior direito podem ser considerados "liberais": eles predizem a classe positiva com maior freqüência, de tal maneira que eles classificam a maioria dos exemplos positivos corretamente, mas, geralmente, com altas taxas de falsos positivos.

A linha diagonal ascendente $(0, 0)$ – $(100\%, 100\%)$ representa um modelo de comportamento estocástico: cada ponto $(p, p)$ pode ser obtido pela previsão da classe positiva com probabilidade $p$ e da classe negativa com probabilidade $100\% - p$. Pontos pertencentes ao triângulo superior esquerdo a essa diagonal representam modelos que desempenham melhor que o aleatório e pontos pertencentes ao triângulo inferior direito representam modelos piores que o aleatório. A diagonal descendente $(0, 100\%)$ – $(100\%, 0)$ representa modelos de classificação que desempenham igualmente em ambas as classes ($tpr = 1 - fpr = tnr$). À esquerda dessa linha estão os modelos que desempenham melhor para a classe negativa em detrimento da positiva e, à direita, os modelos que desempenham melhor para a classe positiva. O espaço ROC está representado esquematicamente na Figura 4.1.

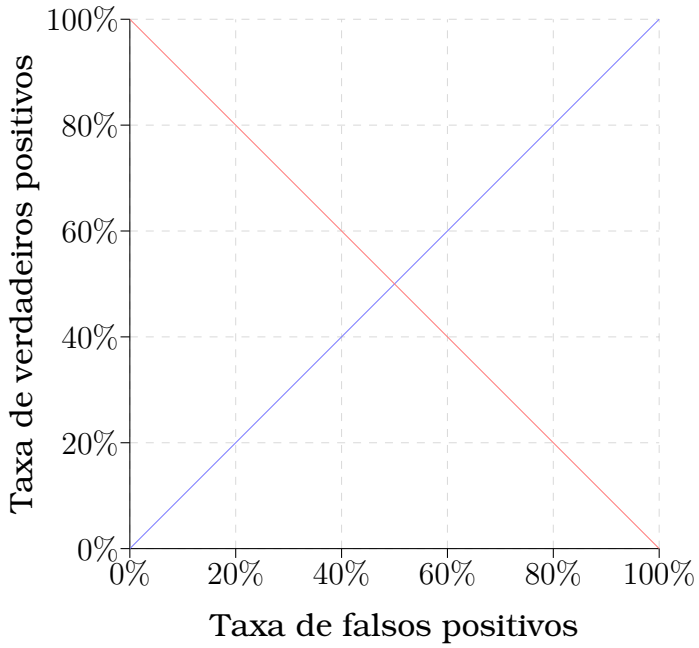
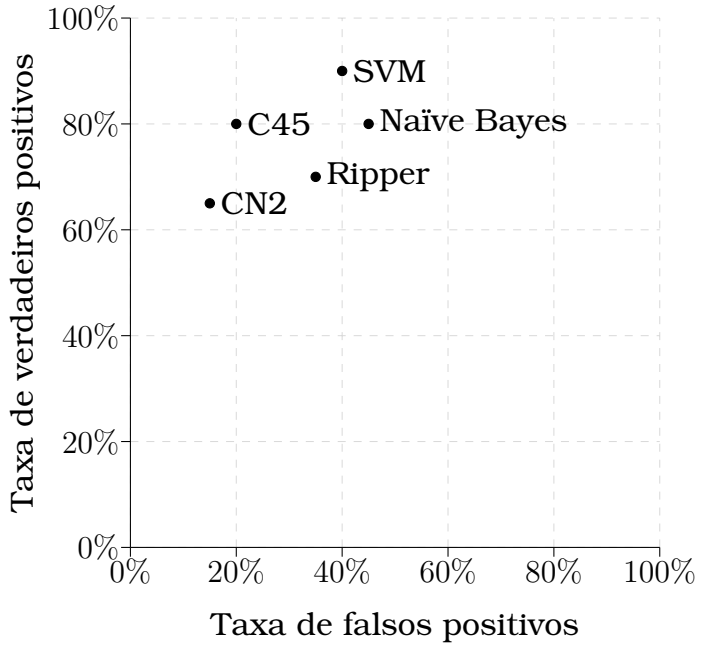


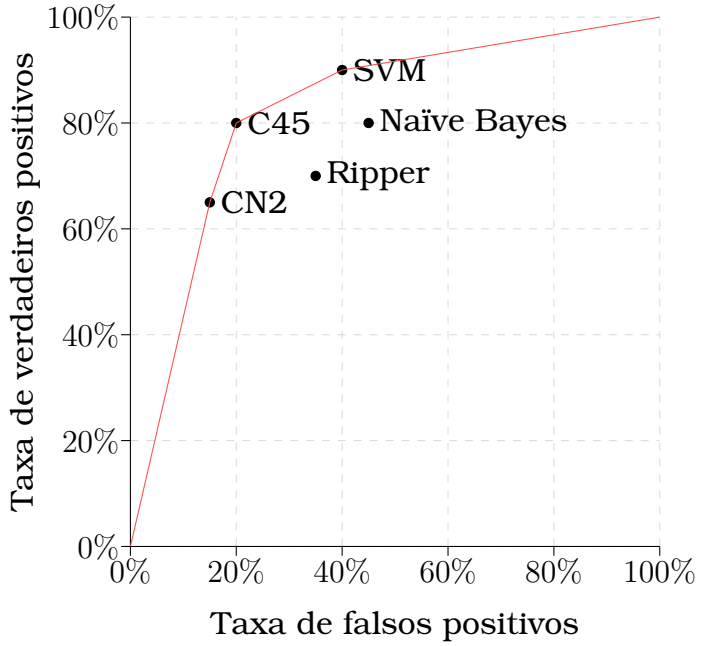

**Figura 4.1:** Espaço ROC

Na Figura 4.2(a) é mostrado um gráfico ROC com 5 modelos de classificação induzidos por 5 sistemas de aprendizado diferentes (C4.5, CN2, Ripper,

Support Vector Machine – SVM, e Naïve Bayes). Uma rápida inspeção visual permite algumas constatações. Neste caso, CN2 é o mais conservativo e SVM é o mais liberal. Além disso, é fácil de perceber que um ponto no espaço ROC é melhor que outro se e somente se ele está acima e a esquerda do outro ponto, *i.e.*, tem uma maior taxa de verdadeiros positivos e uma menor taxa de falsos positivos (Fawcett, 2003).




(a) Modelos de classificação no espaço ROC




(b) Região convexa

**Figura 4.2:** Modelos de classificação no espaço ROC

Além disso, é possível mostrar que os modelos que se encontram na região convexa que mais se aproxima ao ponto $(0, 1)$, como mostrado na

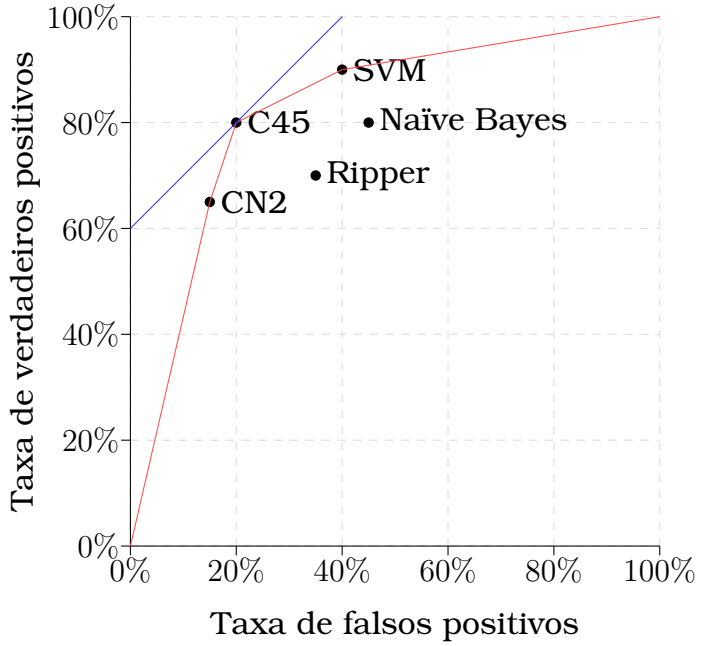
Figura 4.2(b), são os modelos que podem ser considerados ótimos, dada uma certa condição operacional[4]. Os outros modelos que não fazem parte da região convexa podem ser descartados (Provost et al., 1998; Provost & Fawcett, 2001). Isso se deve ao fato de que cada condição operacional define a inclinação de uma linha no espaço ROC, chamada de linha de *iso-desempenho*. Ela recebe esse nome pois todos os pontos que fazem parte dessa linha têm uma característica em comum: a taxa de erro (ou custo médio esperado) é a mesma. A inclinação desta linha está relacionada ao quanto um determinado erro é relativamente mais importante que o outro. O modelo ótimo para uma dada condição operacional deve estar em uma linha com essa inclinação. Além disso, o modelo ótimo deve estar o mais próximo possível do ponto $(0, 100\%)$. Essas duas propriedades implicam que os modelos ótimos estejam na região convexa — uma prova detalhada pode ser encontrada em (Provost & Fawcett, 2001).

Na Figura 4.3 é mostrado o mesmo gráfico ROC da Figura 4.2, com a adição de diferentes linhas de iso-desempenho. A inclinação da linha de iso-desempenho mostrada na Figura 4.3(a) é 1. Isso é equivalente a dizer que se essa linha representar a condição operacional real à proporção de exemplos entre as classes (ou o custo de classificar erradamente um exemplo positivo ou negativo) são iguais. Nessas condições, o modelo induzido pelo C4.5 irá apresentar a menor taxa de erro (ou o menor custo de classificação). Já para o gráfico mostrado na Figura 4.3(b), a inclinação da curva é de 0,5. Se essa linha representar as condições operacionais verdadeiras, a classe positiva é duas vezes mais populosa (ou o custo de classificar erradamente um exemplo da classe positiva é duas vezes maior) que a classe negativa. Nessas condições, ambos os modelos induzidos pelo C4.5 e SVM são ótimos, *i.e.*, tem a mesma taxa de erro/custo de classificação global. Note que, no entanto, as taxas de erros separadas por classes são diferentes para cada um desses algoritmos.

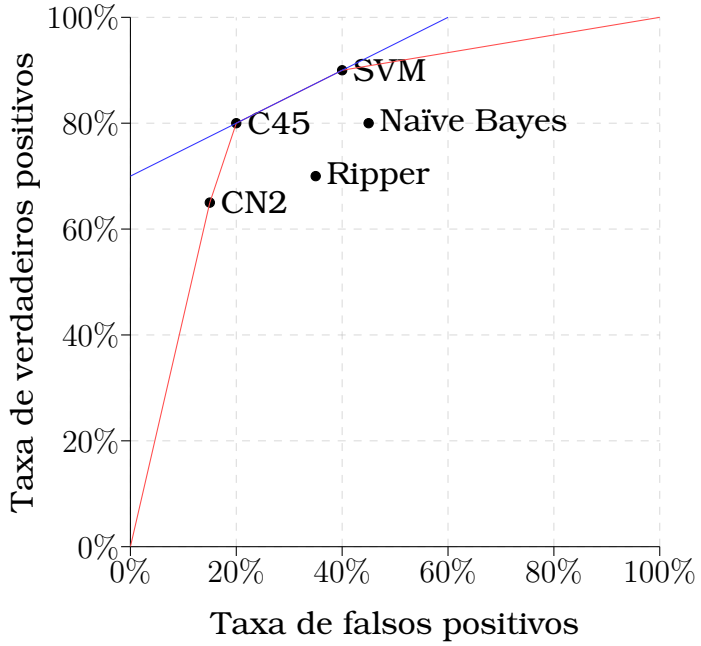
Uma outra vantagem de se utilizar a curva ROC está na avaliação da ordenação dos exemplos, ao invés da classificação. Nesse caso, como descrito na Seção 4.1.2, o sistema de aprendizagem não prediz a classe, mas um valor contínuo ou ordinal, e, para se criar o modelo de classificação, esse valor contínuo pode ser binarizado pela escolha de um limiar de classificação. Entretanto, quando se estabelece um limiar específico, assume-se um certo compromisso entre os acertos e os erros. Esse compromisso pode ser avaliado utilizando a análise de curvas ROC.

Ao invés de se escolher um limiar arbitrário e representar o desempenho do sistema para um dado domínio como um único ponto no espaço ROC,

[4]O termo condição operacional engloba fatores como proporção de exemplos *a priori* entre as classes e/ou custos/benefícios de classificação.




(a) Custos iguais para exemplos positivos e negativos




(b) A classe positiva é duas vezes mais custosa que a negativa

**Figura 4.3:** Diferentes linhas de iso-desempenho implicam em diferentes modelos ótimos

pode-se "simular" a escolha de vários limiares. Nesse caso, varia-se o limiar em todo o seu espectro, desde o valor mais restritivo até o valor mais liberal. O desempenho do sistema é então representado por uma curva no espaço ROC — a curva ROC. Dessa maneira, a análise é feita independentemente da escolha do limiar. Quanto mais distante a curva estiver da diagonal principal, melhor será o desempenho do sistema de aprendizado

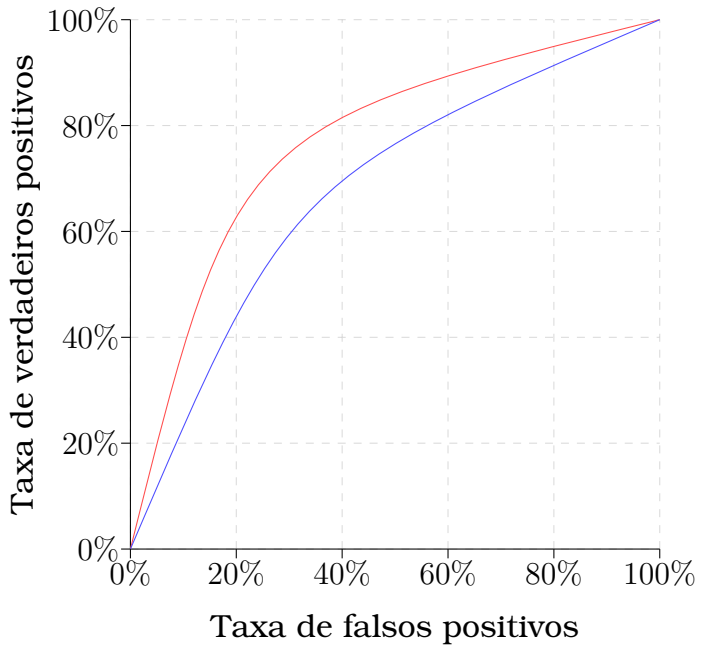
para aquele domínio. Ao se comparar duas (ou mais) curvas, caso não haja nenhuma interseção, a curva que mais se aproxima do ponto $(0, 100\%)$ é a de melhor desempenho. Caso haja intersecções, cada um dos sistemas tem uma faixa operacional na qual é melhor que o outro. Idealmente, a curva deve ser convexa e sempre crescente.

Na Figura 4.4 são mostrados alguns exemplos de curvas ROC. Note que na Figura 4.4(a) não há intersecções entre as curvas. Nesse caso, modelos de classificação derivados a partir da curva mais próxima do ponto $(0, 1)$ serão sempre melhores do que os modelos derivados a partir da outra curva, independentemente das condições operacionais. Já no caso da Figura 4.4(b), em que há uma interseção entre as curvas perto do ponto $(40\%, 70\%)$, os medelos de classificação derivados à esquerda deste ponto serão os melhores se forem derivados da curva que sobe mais próxima do eixo $y$, e da outra curva se forem derivados à direita.

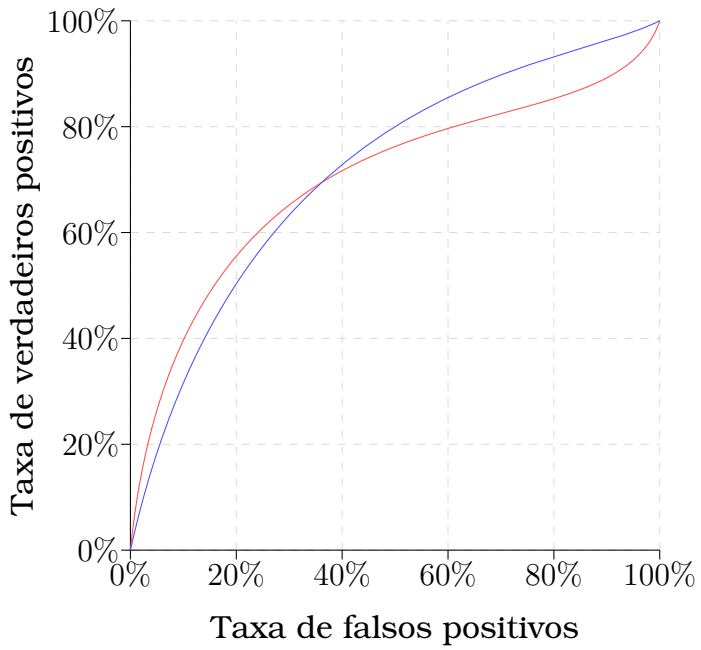
Caso um único modelo seja realmente necessário, a sua derivação pode ser feita com base na análise da curva ROC, analisando-se os possíveis compromissos entre as classificações positivas e negativas. A idéia básica é a mesma das linhas de iso-desempenho. Dada uma condição operacional (e por conseqüência uma linha de iso-desempenho), é possível derivar o limiar apropriado para aquela condição operacional. Dessa maneira, é possível calibrar o modelo para as condições operacionais reais, ao invés daquelas utilizadas durante a indução. Esse fato é o segundo grande chamativo da análise ROC.

Em suma, a primeira vantagem da análise ROC é que pode-se fazer uma análise independentemente de certas condições, tais como o limiar de classificação, os custos relacionados à classificações errôneas e à distribuição *a priori* das classes. Essa análise é realizada visualizando-se o compromisso entre *tpr* e *fpr* em um gráfico bi-dimensional. A segunda vantagem é que a análise ROC pode ser utilizada para a calibração e ajuste de modelos de classificação, quando necessário.

Essas duas propriedades são às vezes mal interpretadas. É comum encontrar afirmativas de que a análise ROC é independente da distribuição de exemplos entre as classes. Na realidade, essa não é uma característica da análise ROC em si, mas uma conseqüência da aplicação das leis básicas da probabilidade citadas na Seção 4.1.1. Ela somente é verdadeira se, como dito na Seção 4.1.1, assumirmos a distribuição de $X$ como característica do domínio. A análise ROC é sim invariante a proporção de exemplos entre as classes desde que essa assumpção seja verdadeira. Entretanto, essa propriedade não é válida em casos em que essa assumpção não é válida. Muitas vezes, essa confusão é empregada como uma crítica a análise ROC (Webb &




(a) Curva sem interseção




(b) Curva com interseção

**Figura 4.4:** Exemplos de curvas ROC

Ting, 2005). No entanto, essa má interpretação está relacionada muito mais com a metodologia (decorrente de considerar verdadeira uma falsa assumpção) do que no método. Quando propriamente utilizada, a análise ROC é uma ferramenta poderosa para a avaliação de sistemas, principalmente em domínios para os quais não se pode definir as condições operacionais com precisão (Fawcett & Flach, 2005).

Uma outra conexão entre a curva ROC e a ordenação dos exemplos está relacionada com a área abaixo da curva ROC — AUC (do inglês *Area Under*

*Curve*). Uma vez que a área abaixo da curva ROC é uma fração da área de um quadrado de lado um, o seu valor está sempre entre 0 e 1. Hanley & McNeil (1982) mostraram que essa área é numericamente equivalente à estatística de Wilcoxon. Além disso, AUC também é numericamente igual a probabilidade de, dados dois exemplos de classes distintas, o exemplo positivo seja ordenado primeiramente que um exemplo negativo (Egan, 1975). AUC também está correlacionada com o coeficiente Gini (Hand & Till, 2001).

A AUC vem gradativamente ganhando espaço como medida de avaliação de modelos em aprendizado de máquina e mineração de dados. Apesar de, como discutida na Seção 4.1.2, a avaliação de um modelo por uma única medida não ser a mais apropriada, AUC tem menos deficiências do que a taxa de erro de classificação (Ling et al., 2003). Entretanto, sempre que possível, é recomendável plotar e analisar a curva dos modelos.

Também é importante ressaltar que, assim como a taxa de erro de classificação, ambos AUC e a curva ROC são variáveis aleatórias e, portanto, devem ser estimadas e acompanhadas de alguma medida de variação. AUC é normalmente estimada da mesma maneira que a taxa de erro (normalmente utilizando-se validação cruzada). Estimar a curva em si é um pouco mais complexo, uma vez que ela é bi-variada. Nesse caso, pode-se calcular tanto a variância com relação a *tpr*, *fpr*, ou ambos. Fawcett (2003) apresenta uma descrição desses três métodos para se estimar a curva média. Na Figura 4.5 é mostrada uma curva ROC com estimava de variância feita com o pacote ROCR[5], desenvolvida para o software R[6]. A linha contínua é a média das linhas pontilhadas, na qual a média foi calculada com relação à *tpr*. As barras de erro indicam intervalos de 1 desvio padrão.

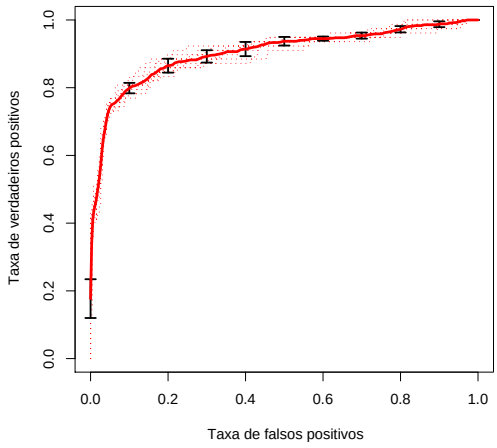



**Figura 4.5:** A curva ROC com variância

[5] http://bioinf.mpi-sb.mpg.de/projects/rocr/
[6] www.r-project.org

Uma das principais desvantagens de se utilizar gráficos ROC é a sua limitação para apenas duas classes. Apesar dos princípios básicos serem os mesmos, o número de eixos cresce exponencialmente com o número de classes. No entanto, algumas aproximações são possíveis, como a um-contra-todos, proposta por Hand & Till (2001).

## 4.3. Considerações finais

Gráficos ROC constituem uma ferramenta muito útil para a visualização e avaliação de modelos de classificação. Gráficos ROC também são utilizados para se avaliar como um sistema de aprendizado é capaz de ordenar os exemplos, permitindo uma análise independente do limiar de classificação. Caso seja necessário derivar um modelo de classificação, a análise ROC também permite que seja feita a calibração, por meio de linhas de iso-desempenho, para as condições operacionais mais apropriadas ao domínio da aplicação.

A análise ROC provê uma avaliação mais rica do que simplesmente avaliar o modelo de classificação a partir de uma única medida. Entretanto, para a sua correta utilização, é necessário que se conheçam suas características e limitações. Neste capítulo foi feita uma introdução à análise ROC dentro do contexto de aprendizado de máquina, bem como foram destacadas e exemplificadas algumas de suas limitações e erros de interpretação.

Capítulo 5

# Novas abordagens para a geração de regras

Neste capítulo são descritos os trabalhos desenvolvidos relacionados à aprendizagem de regras. Na Seção 5.1 é aprensentada uma abordagem para a extração de exceções a partir de regras gerais. Na Seção 5.2 é apresentado o algoritmo ROCCER, um algoritmo que faz seleção de regras baseado em análise ROC. Finalmente, na Seção 5.3, são apresentadas as considerações finais deste capítulo.

## 5.1. Extraindo exceções de regras gerais

Algoritmos clássicos de indução de regras são projetados principalmente para induzir conjuntos de regras com objetivos de classificação ou predição, cuja função é prever ou classificar novos casos com uma alta precisão. Em outras palavras, esses algoritmos tendem a induzir regras com um alto valor de suporte e precisão, de maneira que essas regras induzidas possam formar um modelo para a classificação que apresente uma boa precisão. Um dos problemas com essa abordagem é que muitas das regras induzidas pelos algoritmos de AM podem ser regras tanto triviais quanto difíceis de entender, pois esses algoritmos usam heurísticas e preferências independentes do domínio da aplicação em questão para induzir as regras que compõem o modelo.

A abordagem tradicionalmente utilizada para descobrir conhecimento novo é tratar as regras que compõem o modelo individualmente, filtrando o conjunto de regras e selecionando apenas as mais interessantes segundo algum critério pré-estabelecido (Freitas, 1999). Entretanto, por terem sido induzidas com o objetivo de formarem um modelo, muitas dessas regras são regras gerais, que na maioria das vezes expressam um conhecimento de senso comum. Mesmo que regras gerais sejam usualmente consistentes com as expectativas dos especialistas, em certas atividades de mineração de dados pode ser mais interessante procurar por outros tipos de regras além das gerais.

Um outro ponto a ser considerado é quanto à interpretação e o entendimento das regras induzidas, como visto no Capítulo 3. Os conjuntos de regras induzidos pelos algoritmos de AM podem ser tanto de regras com sobreposição quanto de regras disjuntas. Regras com sobreposição podem ser tanto ordenadas (listas de decisão) quanto não-ordenadas (regras independentes) (Prati et al., 2002).

Listas de decisão impõem uma ordem no conjunto de regras. Do ponto de vista da extração e análise de conhecimento, uma regra extraída de uma lista de decisão é difícil de ser entendida pelo especialista, pois ela é significativa somente no contexto de todas as regras precedentes. Em contrapartida, as regras extraídas de conjuntos não-ordenados ou disjuntos podem ser analisadas individualmente, sendo alternativas mais viáveis do que listas de decisão. Porém, as regras desses conjuntos não têm nenhuma relação entre si. Em muitas tarefas de MD, um relacionamento entre essas regras é muito importante para o entendimento global das relações implícitas do domínio. Em contrapartida, o uso de regras de conhecimento isoladas pode dificultar um entendimento global do domínio.

Como pode ser observado, esses fatores implicam em uma dificuldade na utilização de algoritmos de aprendizado em muitas aplicações práticas de mineração de dados. Essa dificuldade reside tanto na busca de regras quanto na forma de representação utilizadas pelos algoritmos de aprendizado.

### 5.1.1 Exceções

Uma das principais características de regras gerais é que essas regras apresentam exceções (Kivinen et al., 1994). Intuitivamente, exceções contradizem uma regra mais geral ou de senso comum. Também é geralmente aceito que exceções têm um pequeno suporte, mas têm uma confiança similar às regras de senso comum (Suzuki, 1997). Como descrito na Seção 3.6, Suporte é uma medida do número (relativo) de exemplos cobertos correta-

mente pela regra, enquanto que confiança é uma medida do quanto uma regra é precisa. Neste trabalho, usamos o conceito de exceção apresentado em (Hussain et al., 2000), que estruturalmente define exceções como mostrado na Tabela 5.1, na qual $B'$ também representa disjunções não vazias de restrições sobre os atributos.

| | |
|---|---|
| $B \rightarrow H$ | regra geral |
| | alto suporte, alta confiança |
| $B \wedge B' \rightarrow \neg H$ | regra de exceção |
| | baixo suporte, alta confiança |
| $B' \rightarrow \neg H$ | regra de referência |
| | baixo suporte e/ou baixa confiança |

**Tabela 5.1:** Estrutura de regra com exceções

Por exemplo, se houver uma regra de senso comum "*se a pessoa está desempregada não lhe é dado crédito*", uma exceção seria "*se a pessoa está desempregada mas o cônjuge está empregado, então lhe é dado crédito*".

Um ponto importante sobre exceções é que elas podem auxiliar no problema do entendimento do conhecimento induzido. Um conjunto de regras isoladas é pouco intuitivo para o especialista, pois as pessoas geralmente expressam conhecimento em termos de padrões gerais e casos especiais. O conceito implícito de localidade (uma exceção é aplicada somente se a sua regra geral é válida) torna o par (regra geral, regras de referência), que formam as exceções, mais confortável para as necessidades do especialista: regras gerais são expressas primeiro, e exceções à essas regras são modeladas como um posterior refinamento da hipótese.

Um outro ponto importante à respeito de exceções refere-se ao problema da indução de regras interessantes ou úteis. Por contradizerem regras de senso comum, exceções são geralmente mais interessantes e mais úteis para o usuário. Exceções podem, por exemplo, representar um importante nicho em um determinado mercado de consumidores. Reconhecer esse nicho como uma exceção pode conduzir a uma campanha de *marketing* específica à esses consumidores.

Para ilustrar o conceito, considere o conjunto de exemplos representado na Figura 5.1. Esse conjunto contém exemplos descritos por dois atributos, $A1$ e $A2$ e duas classes, ▲ e ○. O domínio de $A1$ é $\{a, b, c, d\}$ e o domínio de $A2$ é $\{x, y\}$. Algoritmos de indução de regras ou generalizam, formulando uma hipótese do tipo $A2 = x \rightarrow$ ▲ e $A2 = y \rightarrow$ ○, ou especificam, formulando uma hipótese para cada exemplo do tipo $A2 = x \wedge A1 = a \rightarrow$ ▲, e assim por diante.

Entretanto, uma maneira mais intuitiva de descrever o conceito é generalizar, permitindo que primeiro se tenha uma idéia geral do conceito, e

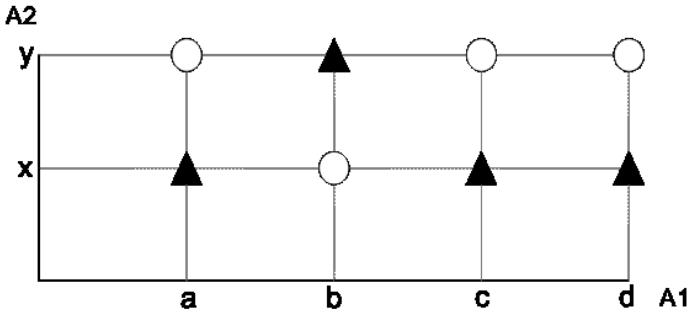



**Figura 5.1:** O espaço de dados de exemplo

depois especificar localmente, tratando exceções e casos especiais. Nesse caso, a hipótese seria

- $A2 = x \rightarrow \blacktriangle$ exceto se $A1 = b \rightarrow \circ$ e
- $A2 = y \rightarrow \circ$ exceto se $A1 = b \rightarrow \blacktriangle$.

### 5.1.2 O método proposto

Diversos métodos foram propostos na literatura para extrair exceções (Suzuki, 1997; Liu et al., 1999; Hussain et al., 2000; Kivinen et al., 1994). Entretanto, esses métodos são geralmente voltados para geração de regras de associação, e poucos desses métodos podem ser aplicados para extrair exceções a partir de regras de classificação. A seguir é descrito um método para encontrar exceções a partir de regras de classificação. Essa abordagem baseia-se nos seguintes princípios:

1. Um algoritmo de aprendizado pode sumarizar dados e induzir regras;
2. Algoritmos de aprendizado geralmente dão preferência à indução de regras com alto suporte;
3. Exceções devem ter um suporte pequeno em toda a amostra de dados, caso contrário seria uma regra de senso comum;

Os três princípios acima implicam em uma dificuldade em encontrar exceções a partir de algoritmos de aprendizado de máquina tradicionais. Extrair exceções diretamente a partir dos dados não é uma tarefa fácil, pois, como mencionado anteriormente, os algoritmos de aprendizado têm heurísticas e preferências que privilegiam regras gerais. Mesmo que essas preferências sejam relaxadas, o conhecimento induzido é fragmentado no conjunto de regras, perdendo-se o conceito de localidade da exceção. Assim, considerando essas dificuldades, a abordagem proposta neste trabalho para encontrar exceções considera duas etapas, descritas a seguir:

**Etapa 1 — indução de regras de senso comum.** Nessa etapa, é utilizado um algoritmo de aprendizado de máquina tradicional para a indução de regras de classificação. Normalmente, o usuário pára aqui a atividade de extração de regras, avaliando e analisando as regras induzidas. Na abordagem proposta, estamos especialmente interessados na indução de regras gerais ou de senso comum. Em outras palavras, a idéia é não permitir, nesta etapa, a indução de regras muito especializadas. Isso é conseguido manipulando apropriadamente os parâmetros do algoritmo de aprendizado.

Após, cada uma das regras $B \rightarrow H$ presentes no conjunto de regras que constituem o modelo de classificação é avaliada utilizando um conjunto de $n_{ex}$ exemplos para construir a sua matriz de contingência. Como descrito na Seção 3.6, a matriz de contingência computa, para cada regra, os valores para os quais $B$ e $H$ são verdadeiros, $B$ e $H$ são falsos, $B$ é verdadeiro e $H$ é falso e $B$ é falso e $H$ é verdadeiro. Para facilitar a leitura, na Tabela 5.2 é mostrada novamente a matriz de contingência, mas diferentemente da matriz apresentada na Tabela 3.3, na qual os valores foram apresentados como valores absolutos, na Tabela 5.2 cada entrada $f_x$ representa a freqüência relativa do evento $x$. Essa matriz é utilizada no passo seguinte para selecionar as regras que serão consideradas para procurar por exceções.

| | $H$ | $\overline{H}$ | |
|---|---|---|---|
| $B$ | $f_{bh}$ | $f_{b\overline{h}}$ | $f_b$ |
| $\overline{B}$ | $f_{\overline{b}h}$ | $f_{\overline{bh}}$ | $f_{\overline{b}}$ |
| | $f_h$ | $f_{\overline{h}}$ | 1 |

**Tabela 5.2:** Matriz de contingência com freqüências relativas para uma regra $B \rightarrow H$

**Etapa 2 — procura por exceções.** Nessa etapa, regras com alto valor de $f_{b\overline{h}}$ são utilizadas na busca por exceções. Um alto valor de $f_{b\overline{h}}$ indica que existe uma certa quantidade de exemplos que satisfazem o corpo $B$ da regra mas que não estão de acordo com a previsão da sua cabeça $H$. Esse valor indica que podem haver exceções para essa regra. Somente esses exemplos são selecionados para compor um novo subconjunto de exemplos no qual as exceções serão procuradas.

Dentro desse subconjunto, associações entre os atributos desse exemplo e a(s) classe(s) negativa(s) $\overline{H}$[1] são procuradas nesse subconjunto.

[1] No caso de existirem mais de duas classes, o conjunto das classes não previstas por $H$ constituem as classes negativas $\overline{H}$.

Se essas associações têm valores de suporte e confiança aceitáveis dentro do subconjunto, elas são regras de referência, e o par (regra geral, regras de referência) representa uma regra de exceção.

Deve ser ressaltado que mesmo que a regra de referência tenha um alto suporte dentro do subconjunto de exemplos utilizado para procurar por exceções, elas têm um suporte pequeno em todo o conjunto de exemplos. Essa abordagem, além de permitir a extração de exceções a partir de regras de classificação, também preserva a localidade da exceção.

### 5.1.3 Estudo de caso

Para testar o método proposto, foi escolhido um conjunto de dados reais, referente à clivagem de proteínas virais por proteases virais para o vírus da AIDS (HIV). Esse conjunto de dados foi também utilizado em Narayanan et al. (2002). Na Tabela 5.3 são apresentadas as principais características do conjunto de dados HIV[2].

| $n_{atr}$ | $n_{ex}$ | valores desconhecidos | classes |
|---|---|---|---|
| 8 (discretos) | 362 | não | 0 - non-cleavage (68,51%)<br>1 - cleavage (31,49%) |

**Tabela 5.3:** Descrição do conjunto de dados HIV utilizado para procurar exceções

A protease viral desempenha um papel importante no ciclo de vida viral, uma vez que é responsável por quebrar a poliproteína (o substrato) em sítios específicos, a medida que essa poliproteína deixa o ribossomo da célula hospedeira como uma seqüência longa. Em outras palavras, a protease é responsável pela pós-tradução de cadeias peptídicas em proteínas e enzimas virais que criarão novos vírus. Deve ser ressaltado que um bom entendimento desse processo é de fundamental importância no estudo de drogas que possam vir a inibir esse processo de clivagem.

O conjunto de exemplos é formado por subseqüencias de aminioácidos de tamanho 8 e da informação (classe) relacionada à subseqüencia representar ou não um sítio de clivagem. O tamanho da subseqüência corresponde ao tamanho da seqüencia de aminoácidos da protease que se liga ao substrato, em um esquema "chave-e-fechadura", ou seja, quando a seqüencia da protease casa com a seqüência do substrato ocorre a clivagem, e viceversa. No caso do HIV, a clivagem ocorre entre a posição 4 e a posição 5 da seqüência.

[2]Disponível em `http://www.dcs.ex.ac.uk/~anarayan/ismbdatasets`

Resultados de experimentos realizados com esse mesmo conjunto de dados, utilizando validação cruzada com 10 partições, uma rede neural MLP treinada com o algoritmo *backpropagation* do simulador SNNS[3] e o indutor de árvores de decisão See5[4] estão reportados em (Narayanan et al., 2002), e são transcritos na Tabela 5.4, na qual os valores entre parênteses referem-se ao desvio-padrão.

| | Rede Neural | See5 |
|---|---|---|
| erro | $7,50(2,76)$ | $14,53(3,31)$ |
| especificidade | $90,80(5,90)$ | $90,20(5,12)$ |
| sensitividade | $82,19(6,94)$ | $75,27(7,65)$ |

**Tabela 5.4:** Taxas de erro de uma rede neural e do See5 no conjunto de dados HIV. Resultados reportados em Narayanan et al. (2002)

Para verificar a possível ocorrência de um superajustamento das exceções sobre o conjunto de dados, a metodologia proposta para encontrar exceções foi repetida três vezes com amostragens diferentes de treinamento e teste. O conjunto de dados foi particionado em três subconjuntos, e, em cada aplicação, duas partições foram usadas para treinamento e a outra para teste.

Para a aplicação da primeira etapa do método proposto, foi utilizado o programa para induzir árvores de decisão See5, com a opção de gerar subconjuntos de valores nos nós de decisão. Como o número de regras foi pequeno em cada execução, todas as regras induzidas na primeira etapa foram selecionadas para a aplicação da segunda etapa.

Na segunda etapa, cada uma das regras define o subconjunto de dados a ser utilizado para procurar por exceções. Como mencionado, esse subconjunto é composto pelos exemplos cobertos pela regra mas que não são da mesma classe por ela prevista. Nesses subconjuntos foi aplicado o algoritmo de geração de regras de associação *Apriori* (Agrawal et al., 1993). Somente as regras que apresentam associações entre os atributos e a classe negativa, e com suporte maior que 0,5 foram selecionadas como possíveis exceções. Se a exceção apareceu em regras semelhantes em pelo menos 2 das 3 execuções, essa exceção foi assumida como válida. As regras transcritas da árvore de decisão na primeira execução são mostradas na Tabela 5.5. Os números entre parênteses indicam, respectivamente, o número de exemplos cobertos e o número de exemplos cobertos incorretamente pela regra.

Dos 149 exemplos cobertos pela regra R1.1 no conjunto de treinamento, 16 pertencem à classe cleavage. Nesse subconjunto de 16 exemplos, foi aplicado o algoritmo de geração de regras de associação. Apenas uma regra

[3] `http://www-ra.informatik.uni-tuebingen.de/SNNS/`
[4] `http://www.rulequest.com`

| | |
|---|---|
| R1.1 | **if** pos4 ∈ {A,R,N,D,C,Q,E,G,H,I,K,P,S,T,W,V}<br>**then** non-cleavage (149,16) |
| R2.1 | **if** pos4 ∈ {L,M,F,Y} **and** pos5 ∈ {A,Q,G,K}<br>**then** non-cleavage (12,3) |
| R3.1 | **if** pos4 ∈ {L,M,F,Y} **and** pos5 ∈ {R,N,D,C,E,H,I,L,M,F,P,S,T,W,Y,V}<br>**then** cleavage (81,15) |

**Tabela 5.5:** Transcrição em regras da árvore de decisão induzida pelo See5 – Execução 1

com os requisitos exigidos foi gerada: **if** pos6 = E **then** cleavage, que cobre 9 exemplos desse subconjunto. No conjunto de teste, essa exceção cobriu corretamente o único exemplo que era coberto erroneamente pela regra, mas outros dois exemplos cobertos corretamente passaram a ser erroneamente cobertos.

O mesmo procedimento foi aplicado à regra R2.1 e R3.1. Para a regra R2.1, 3 exemplos pertencem à classe cleavage. Para essa regra, também foi gerada pelo *Apriori* a regra: **if** pos6 = E **then** cleavage, que cobre todos os 3 exemplos no conjunto de treinamento. No conjunto de teste, dos 7 exemplos cobertos por essa regra, 5 foram erroneamente cobertos pela regra R2.1. A exceção passou a cobrir corretamente 3 desses 5 exemplos. Para a regra R3.1, dos 15 exemplos cobertos erroneamente no conjunto de treinamento, não foram encontradas associações que satisfizessem o critério mínimo de suporte.

As regras extraídas da árvore de decisão na segunda execução são mostradas na Tabela 5.6. Dos 158 exemplos cobertos pela regra R1.2, 15 pertencem à classe cleavage. Nesse subconjunto de exemplos foi aplicado o algoritmo de geração de regras de associação e a mesma regra (**if** pos6 = E **then** cleavage) foi encontrada. Dos 15 exemplos do subconjunto, 8 são cobertos pela regra. No conjunto de teste, a exceção passou a cobrir corretamente todos os 5 exemplos que eram erroneamente cobertos, e passou a cobrir erroneamente 1 exemplo que era coberto corretamente pela regra R1.2. Nenhuma exceção que satisfizesse os critérios estabelecidos foi encontrada para a regra R2.2.

| | |
|---|---|
| R1.2 | **if** pos4 ∈ {A,R,N,D,C,Q,E,G,H,I,K,M,P,S,T,W,V}<br>**then** non-cleavage (158,15) |
| R2.2 | **if** pos4 ∈ {L,F,Y}<br>**then** cleavage (84,22) |

**Tabela 5.6:** Transcrição em regras da árvore de decisão induzida pelo See5 – Execução 2

As regras extraídas da árvore de decisão na terceira execução são mostradas na Tabela 5.7. Dos 153 exemplos cobertos pela regra R1.3, 12 pertencem à classe cleavage. Como nas execuções 1 e 2, nesse subconjunto

de exemplos foi aplicado o algoritmo de geração de regras de associação e a mesma regra (**if** pos6 = E **then** cleavage) foi encontrada. Dos 12 exemplos do subconjunto, 7 foram cobertos pela regra. No conjunto de teste, a exceção passou a cobrir corretamente 3 dos 5 exemplos que eram erroneamente cobertos. Nenhum dos exemplos que foram cobertos corretamente pela regra R1.3 passaram a ser erroneamente cobertos. Nenhuma exceção que satisfizesse os critérios estabelecidos foi encontrada para a regra R2.3.

| | |
|---|---|
| R1.3 | **if** pos4 $\in$ {A,R,N,D,C,Q,E,G,H,I,K,P,S,T,W,V} **then** non-cleavage (153,12) |
| R2.3 | **if** pos4 $\in$ {L,M,F,Y} **then** cleavage (81,15) |

**Tabela 5.7:** Transcrição em regras da árvore de decisão induzida pelo See5 – Execução 3

Para verificar se a taxa de erro média das três execuções obtida pela nossa abordagem é comparável à obtida pelo See5, aplicamos o teste $t - student$ pareado para testar a seguinte hipótese:

$$\begin{cases} H_0: & TaxaErro_{\text{Exceções}} = TaxaErro_{\text{See5}} \\ H_1: & TaxaErro_{\text{Exceções}} \neq TaxaErro_{\text{See5}} \end{cases}$$

A taxa de erro média obtida pelo See5 é de 14,63% com um desvio padrão de 5.95. Utilizando a nossa abordagem, a taxa de erro é de 13,53% com um desvio padrão de 6.24. Com esses valores é possível rejeitar a hipótese $H_0$ com 95% de confiança. Além da taxa de erro, também aplicamos o teste $t - student$ na sensitividade média (taxa de verdadeiros positivos) para testar a seguinte hipótese:

$$\begin{cases} H_0: & Sensitividade_{\text{Exceções}} = Sensitividade_{\text{See5}} \\ H_1: & Sensitividade_{\text{Exceções}} \neq Sensitividade_{\text{See5}} \end{cases}$$

A sensitividade média alcançada pelo See5 foi de 85,09% com 7,60 de desvio padrão. Nossa abordagem teve uma sensitividade média de 93,86% com um desvio padrão de 4.02. Com esses valores também é possível rejeitar a hipótese $H_0$ com 95% de confiança.

As três aplicações da metodologia proposta encontraram a mesma exceção para regras que tinham como previsão non-cleavage. Esse fato fornece um forte indício que ela representa realmente uma exceção, e não uma mera casualidade nos dados. Para potencializar a ação da metodologia proposta na descoberta de conhecimento, uma quarta aplicação foi feita, desta vez com todos os exemplos disponíveis. Na regra R1, mostrada na Tabela 5.8, que é muito similar às regras gerais das três primeiras aplicações, nova-

mente foi encontrada a exceção **if** pos6 = E **then** cleavage. A hipótese final, com as exceções adicionadas à regras R1, é transcrita na Tabela 5.8. Dos 229 exemplos cobertos pela regra R1, 17 foram erroneamente cobertos. Com a adição da exceção à regra R1, 10 desses exemplos passaram a ser corretamente cobertos. No entanto, 6 dos exemplos corretamente cobertos passaram a ser cobertos de maneira errada.

| | |
|---|---|
| R1 | **if** pos4 ∈ {A,R,N,D,C,Q,E,G,H,I,K,P,S,T,W,V}<br>**then** non-cleavage<br>**exception: if** pos6 = E **then** cleavage |
| R2 | **if** pos4 ∈ {L,M,F,Y} **and** pos5 ∈ {A,R,E,G,H,I,L,M,F,P,T,W,Y,V}<br>**then** cleavage |
| R3 | **if** pos4 ∈ {L,M,F,Y} **and** pos5 ∈ {N,D,C,Q,K,S}<br>**then** non-cleavage |

**Tabela 5.8:** Hipótese final com exceções encontrada pela metodologia proposta

As regras gerais confirmam a importância dos aminoácidos na posição 4 e 5 no processo de clivagem do substrato protéico. Esse ponto é justamente o ponto onde ocorre a catálise na ligação que irá ser clivada (ligação *scissile*). As exceções geradas mostram a importância da posição 6 no processo de clivagem. Esse fato também foi destacado em (Narayanan et al., 2002).

Em uma tentativa de capturar as exceções utilizando somente um algoritmo de aprendizado tradicional (no caso, o See5, com a opção de gerar conjuntos de regras e sem a opção de agrupar subconjuntos de valores), relaxamos o fator de confiança até o valor de 60% (o procedimento utilizado foi relaxar o fator padrão, que é 25%, de 5 em 5 pontos percentuais até que o atributo pos6 aparecesse em pelo menos uma regra do conjunto de regras).

Essa configuração resultou em um conjunto com 32 regras. O atributo pos6 = E apareceu na regra 29: **if** pos6 = E **then** cleavage. Entretanto, essa regra deve ser analisada com cuidado, uma vez que o See5 com a opção de gerar conjuntos de regras agrupa suas regras em blocos, de acordo com a classe. Uma regra só tem sentido se nenhuma regra do bloco anterior for disparada. Essa regra 29 pertence ao segundo bloco de regras e, portanto, só é valida se o exemplo em questão não for coberto por nenhuma regra do primeiro bloco. Traçando um paralelo com exceções, as regras do bloco 2 podem ser pensadas como exceções às regras do bloco 1. Entretanto, o conceito de localidade da exceção é perdido, uma vez que as regras são apresentadas em blocos, e uma regra de um bloco não tem nenhuma relação com as dos outros blocos, a não ser no que se refere à ordenação dos blocos.

O mesmo procedimento foi utilizado com o See5, desta vez sem a opção de gerar conjuntos de regras e sem a opção de agrupar subconjuntos de valores. Nesse caso, o resultado é a indução de uma árvore de decisão, que pode ser transcrita em regras disjuntas. Novamente o fator de confiança para o qual apareceu o atributo pos6 em pelo menos um dos nós da árvore foi de 60%. O tamanho da árvore induzida é de 55 nós folhas, que corresponde a 55 regras. Duas dessas 55 regras apresentam o atributo pos6 = E (**if** pos4 = L **and** pos6 = E **then** cleavage e **if** pos4 = M **and** pos6 = E **then** cleavage). Entretanto, perde-se o conceito de localidade da exceção, pois ela fica fragmentada entre as regras do conjunto de regras.

Esses exemplos ilustram a utilidade do método aqui proposto para procurar por exceções, pois além das regras induzidas por algoritmos tradicionais de AM não preservarem o conceito de localidade, ao relaxar-se o fator de confiança pode-se estar super-ajustando o algoritmo ao problema.

## 5.2. Seleção de regras utilizando análise ROC

Como visto no Capítulo 3, algoritmos de indução de regras de classificação podem ser divididos em duas famílias: *dividir-para-conquistar* e *separar-para-conquistar*. As duas famílias têm várias características em comum, entre elas a suposição que o espaço de exemplos contém regiões contínuas de exemplos de uma mesma classe.

Recordando, nos algoritmos da família separar-para-conquistar (Fürnkranz, 1999), o algoritmo de busca geralmente utilizado é um procedimento iterativo de busca em feixe para a cobertura do conjunto de exemplos que, a cada iteração, encontra a melhor regra (de acordo com os critérios de busca estabelecidos pelo algoritmo) e remove os exemplos cobertos. O processo é recursivamente aplicado aos exemplos restantes até que o conjunto de treinamento esteja vazio. Para formar o modelo de classificação, as regras podem ser arranjadas ou em uma lista ordenada de regras (lista de decisão) ou em um conjunto de regras. Nesse último caso, deve haver um critério adicional para decidir a classificação de um exemplo, no caso em que mais de uma regra que predigam classes diferentes sejam disparadas.

Em contrapartida, na família dividir-para-conquistar (Quinlan, 1986), ao invés de se gerar um modelo de classificação que recursivamente separa uma porção dos exemplos, um modelo global é criado por recursivos refinamentos do modelo atual. A cada iteração, o conjunto de exemplos é particionado em conjuntos disjuntos. O resultado é geralmente expresso como uma árvore de decisão que divide completamente o espaço de exemplos em regiões hiper-retangulares não sobrepostas e paralela aos eixos.

Apesar das árvores de decisão serem muito utilizadas na prática, em muitas aplicações é preferível utilizar algoritmos da família separar-para-conquistar. Isso se deve ao fato de que, pela necessidade de criar modelos completos e com regras disjuntas, em domínios complexos, árvores de decisão tendem a serem de grandes dimensões. Em contrapartida, como os algoritmos do tipo separar-para-conquistar podem induzir regras com sobreposição, os conjuntos de regras podem ser mais simples (van den Eijkel, 2003).

Entretanto, um dos problemas com a abordagem de cobertura de conjunto utilizada na maioria dos algoritmos da família separar-para-conquistar é que as regras são induzidas isoladamente, mesmo que elas sejam utilizadas em conjunto com outras regras dentro do modelo de classificação. Além disso, como o número de exemplos é reduzido a cada iteração, cada vez menos exemplos estão disponíveis para a indução de novas regras. Em muitos casos ocorre uma fragmentação do conjunto de treinamento e a indução de regras com um baixo suporte estatístico (Domingos, 1996). Além disso, cada nova regra é construída em completa ignorância das demais e dos exemplos já cobertos. Se uma regra ruim for introduzida no conjunto de regras, não há como procurar por uma regra melhor que cubra aqueles exemplos por ela cobertos *i.e*, não existe *backtracking*.

### 5.2.1 O algoritmo de seleção de regras ROCCER

Para suprir algumas das deficiências da abordagem de cobertura de conjunto desenvolvemos o algoritmo ROCCER (Prati & Flach, 2004, 2005). A abordagem do ROCCER para a geração de regras consiste em utilizar a análise ROC para a seleção de regras que irão compor o modelo. Como descrito no Capítulo 4, gráficos ROC são curvas da porcentagem de exemplos positivos classificados incorretamente — taxa de falsos positivos (*fpr*) — no eixo $x$ e a porcentagem de exemplos positivos corretamente classificados — taxa de verdadeiros positivos (*tpr*) — no eixo $y$. É importante ressaltar que se pode representar em um gráfico ROC apenas uma regra isoladamente, um modelo de classificação formado por conjunto de regras ou apenas parte do modelo, formado por um subconjunto das regras que o compõem.

Para modelos em que a predição está associada não somente a uma classe, mas também a um valor contínuo, de tal maneira que seja possível ajustar o limiar de classificação, é possível obter vários pontos $(fpr_i, tpr_i)$ pela variação do limiar. Conectando-se os pontos obtém-se uma curva no espaço ROC que representa o comportamento do modelo sobre todas as possíveis escolhas do limiar de classificação.

Dentro de contexto de aprendizado de regras, Fürnkranz & Flach (2005) mostram que o aprendizado de regras utilizando a abordagem de cobertura de conjunto pode ser vista como o traço de uma curva no espaço ROC. Para entender o porquê, assuma que tenhamos uma lista de decisão vazia, representada pelo ponto $(0, 0)$ no espaço ROC. Ao adicionarmos uma nova regra $R_1$ à lista de decisão, movemos do ponto inicial $(0, 0)$ para o ponto $(\mathit{fpr}_1, \mathit{tpr}_1)$. Adicionando a regra $R_j$, movemos do ponto $(\mathit{fpr}_{j-1}, \mathit{tpr}_{j-1})$ para o ponto $(\mathit{fpr}_j, \mathit{tpr}_j)$, no qual $\mathit{fpr}_j$ e $\mathit{tpr}_j$ são as *tpr* e *fpr* da lista de decisão parcial contendo todas as regras induzidas até agora, incluindo $R_j$. Uma curvar pode então ser traçada plotando-se todas as listas de decisão parciais $(\mathit{fpr}_j, \mathit{tpr}_j)$, para $j$ variando de 0 ate o número $n$ de regras na lista de decisão final, na ordem em que elas são lidas. Para se completar a curva, podemos imaginar a adição de uma regra padrão que sempre prediz a classe positiva ao final da lista de decisão, conectando o ponto $(\mathit{fpr}_n, \mathit{tpr}_n)$ ao ponto $(1, 1)$.

A nossa abordagem é baseada nesse mecanismo e no fato de que, em modelos em que é possível ajustar o limiar de classificação, os pontos que representam os limiares ótimos para diferentes condições residem na região convexa mais externa à curva ROC (Provost & Fawcett, 2001). Em nossa abordagem, as regras provêm de um extenso conjunto de regras externo (em nossa implementação, utilizamos o algoritmo *Apriori* para a geração de regras de associação, fixando a cabeça da regra em cada possível valor do atributo classe) e aplicamos um passo de seleção de regras baseado na análise ROC. A idéia básica é de somente inserir uma regra na lista de decisão se, e somente se, essa inserção está fora da região convexa com as regras que já foram inseridas na lista de decisão. Em outras palavras, uma regra é inserida somente se ela aumenta a região convexa das regras já existentes na lista de decisão. Caso contário, a regra é descartada. Para se entender melhor o funcionamento do algoritmo, ele será introduzido utilizando um exemplo.

Regras são selecionadas separadamente para cada uma das classes e são armazenadas em uma lista de decisão. Para tratar problemas com várias classes, o procedimento é repetido para cada uma das classes. Para facilitar a explicação, designamos a classe para a qual estamos selecionando regras como **positiva** e como **negativa** (a conjunção de) os exemplos na(s) outra(s) classe(s). Considere ($\mathit{fpr}_{Rj}$, $\mathit{tpr}_{Rj}$) como as taxas de verdadeiros positivos e falsos positivos de uma dada regra $R_j$ no espaço ROC, ($\mathit{tpr}_i$, $\mathit{fpr}_i$) como um ponto $i$ na curva ROC que corresponde ao modelo construído como uma lista de decisão com $i$ regras.

Primeiramente, inicializamos a lista de decisão com a regra $R_{default}$ padrão, que sempre prediz a classe positiva. A região convexa da curva ROC é então formada por 2 pontos, $(0,0)$ e $(1,1)$. Esses dois pontos têm as seguintes interpretações: "ignore a regra padrão (classifique todo mundo como negativo)" ou "use a regra padrão (classifique todo mundo como positivo)". Suponha agora que queiramos inserir uma nova regra $R_1$. Como a região convexa atual da curva ROC é formada somente pelo segmento de reta $(0,0) - (1,1)$, $R_1$ somente será inserido se o ponto correspondente a regra no espaço ROC $(fpr_{R1}, tpr_{R1})$ está acima da região convexa. Assumamos que $R_1$ é inserido. Então, a região convexa atual é atualizada, e passa a conter os pontos $(0,0), (fpr_1, tpr_1), (1,1)$, no qual $fpr_1 = fpr_{R1}$ e $tpr_1 = tpr_{R1}$. Essa inserção é ilustrado na Figura 5.2.

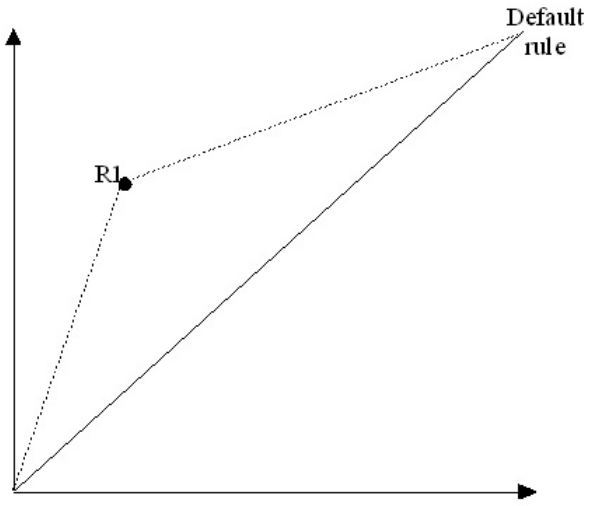
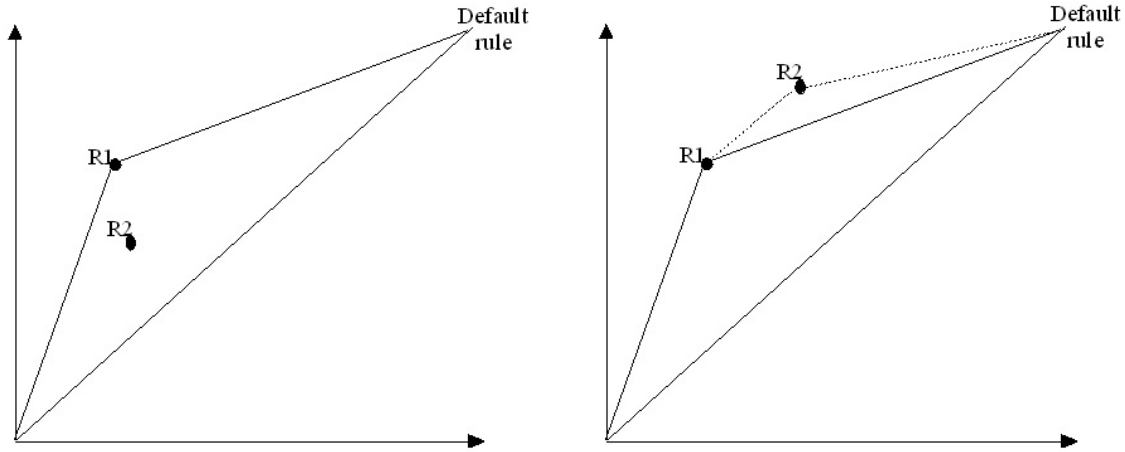



**Figura 5.2:** $R_1$ corresponde a um ponto fora da região convexa (diagonal principal) e é inserida na lista de decisão.

Suponha agora que queiramos inserir uma segunda regra $R_2$. Como dito anteriormente, na abordagem tradicional de cobertura de conjunto o aprendizado de uma nova regra não leva em consideração as regras anteriormente aprendidas. Em nossa abordagem, tentamos contornar essa deficiência utilizado o gráfico ROC para analisar interações entre as regras. Para esse fim, comparamos a regra que estamos tentando inserir com a inclinação de cada segmento de reta na região convexa do gráfico ROC. No exemplo que estamos tratando, se a inclinação do ponto formado pela origem e o ponto $(fpr_{R2}, tpr_{R2})$ está acima da inclinação do segmento formado pela origem e o ponto $(fpr_1, tpr_1)$, $R_2$ está acima da região convexa do gráfico ROC e, portanto, deve ser inserido na lista de decisão antes de $R_1$. Esse fato é equivalente a $R_2$ ter sido induzida antes de $R_1$ na abordagem tradicional de cobertura de conjunto. Uma vez que o algoritmo por nós proposto compara cada nova regra a ser inserida com as regras que compõem a lista de decisão, nosso algoritmo provê um mecanismo implícito de *backtracking*.

Também é importante notar que a inserção de $R_2$ irá provocar mudanças nos outros pontos da curva ROC. O primeiro ponto não trivial do gráfico ROC passará a ser $(fpr_{R2}, tpr_{R2})$. Se as regras $R_1$ e $R_2$ não têm nenhuma sobreposição, o segundo ponto será $(fpr_{R2} + fpr_{R1}, tpr_{R2} + tpr_{R2})$. Entretanto, se houver alguma sobreposição entre $R_1$ e $R_2$, deve-se remover os exemplos cobertos por ambas as regras para se calcular o segundo ponto.

Se $R_2$ não for inserido na primeira iteração, continuamos a comparação com os segmentos de retas remanescentes na região convexa do gráfico ROC. Antes de comparar com o próximo segmento de reta, atualizamos os valores de *fpr* e *tpr* de $R_2$ removendo os exemplos já cobertos por $R_1$ (os exemplos para os quais o antecedente de $R_1$ é verdadeiro). Isso é equivalente a "interpretar" $R_2$ como $\neg R_1 \wedge R_2$. Se a nova posição de $R_2$ está além da região convexa, *i.e.*, a inclinação do segmento de reta formado por $(fpr_1, tpr_1)$ e $(fpr_{R1}, tpr_{R1})$ é maior do que a inclinação do segmento de reta formado por $R_1$ até $R_{default}$, $R_2$ é inserido na lista de decisão depois de $R_1$. Essa inserção é mostrada na Figura 5.3. Caso contrário, uma vez que não existem mais regras na lista de decisão $R_2$, é descartado. O pseudo-código do algoritmo ROCCER é mostrado na Algoritmo 5.1.




(a) $R_2$ não está fora da região convexa quando comparado com $R_1$...

(b) ..mas está quando comparado com a $R_{default}$.

**Figura 5.3:** Procurando pelo ponto correto de inserir $R_2$.

Devido à sobreposição entre as regras, esse procedimento não leva necessariamente a uma curva convexa. A inserção de uma nova regra na lista de decisão pode introduzir concavidades antes ou depois do ponto de inserção. Se uma concavidade ocorrer depois da inserção, a regra inserida cobre exemplos originalmente cobertos por regras subseqüentes. Além disso, a regra inserida cobre com uma melhor precisão os exemplos cobertos pelas regras subseqüentes. Nesse caso, remove-se as regras em que a concavidade ocorre. No caso em que a concavidade ocorrer antes da regra inserida,

**Data**: $RS_{in}$: Um (grande) conjunto inicial de regras de uma dada classe. $E_i = E_i^+ \cup E_i^-$: um conjunto de exemplos positivos e negativos para uma classe $c_i$
**Result**: $RS_{out}$: Um lista de decisão contendo um subconjunto das regras dadas como entrada.
$RS_{out}$ = [$R_{default}$];
**foreach** *Regra* $\in RS_{in}$ **do**
  `/* tenta inserir na lista de decisão */;`
  TENTEINSERIR($Regra$,$RS_{out}$)
**end**
**return** $RS_{out}$

**procedure** TENTEINSERIR($Regra$,$RS_{out}$)
  RegraSerComparada = primeira regra na lista de decisão $RS_{out}$;
  **repeat**
    **if** *(fpr,tpr) de* $Regra$ *está para fora da região convexa* **then**
      Insira $Regra$ em $RS_{out}$ antes de RegraSerComparada
    **else**
      `/* mova para uma nova posição */;`
      Remova todos os exemplos de $Regra$ que são cobertos por RegraSerComparada;
      Atualize (*fpr*,*tpr*) de $Regra$ ;
      RegraSerComparada = próxima regra em $RS_{out}$
    **end**
  **until** *RegraSerComparada* $\neq R_{default}$ ;
  **if** *Regra não é inserida* **then**
    Descarte $Rule$
  **end**

**Algoritmo 5.1**: O algoritmo ROCCER.

tanto a nova regra quanto a regra que já está na lista de decisão compartilham uma região na qual ambas classificam mal uma porção dos exemplos. Nesse caso, é injustificável excluir a regra que já estava na lista de decisão e manter somente a nova regra, pois elas se complementam mutuamente. Nossa abordagem para esse caso é de construir a disjunção das duas regras e tratá-las como uma regra única.

Em nossa implementação, o conjunto inicial de regras apresentado ao ROCCER é inicialmente ordenado utilizando-se a distância ao ponto $(0, 1)$ no espaço ROC. Entretanto, como o ROCCER permite que regras sejam removidas e também provê um mecanismo de *backtracking*, essa ordenação é mais uma conveniência para acelerar a execução do algoritmo. É importante ressaltar que a dependência com relação à ordem em que as regras são induzidas é muito menor do que o algoritmo de cobertura de conjunto, utilizado na maioria dos algoritmos que induzem listas de decisão. Somente em casos de empate com uma regra previamente inserida é que essa pré-ordem é importante, pois a regra que já foi previamente inserida na lista de

decisão será mantida e a nova descartada. No entanto, novos experimentos com diferentes pré-ordenações podem ser interessantes para testar se é possível diminuir o tempo de execução do algoritmo. Concluímos assim a descrição da fase de treinamento do ROCCER.

Para decidir a classe de um novo exemplo, também utilizamos um método baseado em gráficos ROC. O teorema de Bayes afirma que as chances de um modelo de classificação classificar corretamente um novo exemplo (chances *a posteriori*) é dada pela razão da verossimilhança multiplicado pelas chances do novo exemplo ser da classe predita (chances *a priori*). No espaço ROC, a razão de verossimilhança é dada por $\frac{tpr}{fpr}$. Relembre que as regras são selecionadas separadamente para cada uma das classes. Dessa maneira, temos uma lista de decisão e uma região convexa para cada uma das classes. Para classificar um novo exemplo, também consideramos cada classe em separado, e, para cada classe, selecionamos a primeira regra daquela classe que cobre o exemplo (note que se nenhuma regra cobrir o exemplo, a regra $R_{default}$ para aquela classe é disparada) e a sua respectiva inclinação no espaço ROC. Essa inclinação é justamente a razão de verossimilhança. As chances *a posteriori* são então convertidas em probabilidades (para *ranking*) ou selecionamos a classe com maior probabilidade (para classificação).

### 5.2.2 Validação experimental

Para avaliar empiricamente a nossa proposta conduzimos uma avaliação experimental utilizando 16 conjuntos de dados provenientes do repositório de dados da Universidade da Califórnia, campus de Irvine — UCI (Newman et al., 1998). Restringimos os conjuntos de dados a conjuntos sem valores desconhecidos nos atributos pois o algoritmo *Apriori*, utilizado para a geração das regras de associação que usamos como entrada para o ROCCER no experimento, não trata atributos com valores desconhecidos.

Na Tabela 5.9 é mostrado um sumário das principais características dos conjuntos de dados utilizados no experimento. Nessa tabela é mostrado, para cada conjunto de dados, o número de atributos ($n_{atr}$), o número de exemplos ($n_{ex}$), e a porcentagem de exemplos na classe majoritária. Para os conjuntos de dados com mais de duas classes, a classe com menos exemplos foi designada como classe positiva e o restante dos exemplos foram agrupados na classe negativa.

Os resultados obtidos pelo ROCCER foram comparados com os seguintes sistemas de aprendizado de regras:

**CN2** Esse sistema é uma implementação clássica da família de algoritmos separar-para-conquistar. Em sua primeira versão (Clark & Niblett,

| # | Conj. Dados | $n_{atr}$ | $n_{ex}$ | %Classe Maj. |
|---|---|---|---|---|
| 1 | Breast | 9 | 683 | 65.00 |
| 2 | Bupa | 6 | 345 | 57.98 |
| 3 | E.Coli | 7 | 336 | 89.58 |
| 4 | Flag | 28 | 194 | 91.24 |
| 5 | German | 20 | 1000 | 70.00 |
| 6 | Glass | 9 | 214 | 92.07 |
| 7 | Haberman | 3 | 306 | 73.53 |
| 8 | Heart | 13 | 270 | 55.55 |
| 9 | Ionosphere | 33 | 351 | 64.10 |
| 10 | Kr-vs-Kp | 36 | 3196 | 52.22 |
| 11 | Letter-a | 16 | 20000 | 96.06 |
| 12 | New-thyroid | 5 | 215 | 83.72 |
| 13 | Nursery | 8 | 12960 | 97.45 |
| 14 | Pima | 8 | 768 | 65.10 |
| 15 | Satimage | 36 | 6435 | 90.27 |
| 16 | Vehicle | 18 | 846 | 76.48 |

**Tabela 5.9:** Descrição dos conjuntos de dados utilizado no experimento com o ROCCER

1989) o CN2 induz uma lista de decisão utilizando entropia como uma heurística de busca. Ele foi posteriormente modificado para incorporar a indução de conjuntos de regras não ordenadas e a precisão de Laplace como função de avaliação (Clark & Boswell, 1991).

**Ripper** Cohen (1995) propôs o algoritmo Ripper dentro do contexto da *poda por redução incremental do erro* (IREP[5]) (Füurnkranz & Widmer, 1994). Ele tem algumas características como poda baseada em erro e heurísticas baseadas em tamanho mínimo de descrição[6] para determinar quantas regras devem ser aprendidas.

**Slipper** A principal diferença com relação ao Ripper é a utilização de pesos para ponderar os exemplos já cobertos, ao invés de simplesmente removê-los (Cohen & Singer, 1999).

**C4.5** O sistema C4.5 (Quinlan, 1993) é um membro da família de algoritmos dividir-para-conquistar. Ele utiliza o ganho de informação como um medida de qualidade para construir uma árvore de decisão e um sistema de pós-poda baseado na redução do erro. Cada ramo da árvore pode ser considerado como uma regra.

Ripper e Slipper foram utilizados com a opção `-a` habilitada para gerar regras para ambas as classes. CN2 foi utilizado em suas duas versões: com a

[5]do inglês *Incremental Reduced Error Pruning*
[6]*Minimum description lenght*

indução de regras ordenadas (CN2OR) e regras não ordenadas (CN2). Também foram utilizadas ambas as árvores induzidas pelo C4.5: a árvore podada (C45) e a árvore não podada (C45NP). Todos os outros parâmetros foram definidos com seus valores padrão. Para se calcular a área abaixo da curva ROC, probabilidades associadas a cada regra foram calculadas utilizando a correção de Laplace. Na versão não ordenada do CN2, as probabilidades foram calculadas levando-se em consideração todas as regras disparadas. Os valores da AUC foram estimados utilizando a regra do trapézio. Foi utilizada a implementação do *Apriori* implementada por Borgelt & Kruse (2002). Os parâmetros do *Apriori* foram inicializados com 50% de confiança e 1/3 da porcentagem da classe minoritária como suporte. Para o ROCCER, as probabilidades foram estimadas utilizando chances *a posteriori*, como descritos na Seção 5.2.1. Nossa abordagem também foi comparada com um modelo de classificação composto por todas as regras geradas pelo *Apriori*.

Os experimentos foram executados utilizando-se validação cruzada com 10 partições (*10-fold cross validation*). O experimento é pareado, isto é, para todos os sistemas foram fornecidos os mesmos conjuntos de treinamento e teste. A média dos valores da AUC, bem como os respectivos desvios padrão entre parênteses, são mostrados na Tabela 5.10. Para a realização de testes estatísticos foi utilizado o teste bicaudal de Dunnet Hochberg & Tamhane (1987); Hsu (1996), que é indicado para múltiplas comparações com um controle. Para a realização desse teste, ROCCER foi utilizado como controle. Dessa maneira, para cada conjunto de dados, testamos a seguinte hipótese:

$$\begin{cases} H_0: & AUC_{\text{ROCCER}} = AUC_{\text{outros algoritmos}} \\ H_1: & AUC_{\text{ROCCER}} \neq AUC_{\text{outros algoritmos}} \end{cases}$$

O teste de hipótese foi realizado com um nível de confiança de 95%. Na Tabela 5.10, algoritmos cujo desempenho, medidos em termos da área abaixo da curva ROC (AUC), foram estatisticamente melhores que o ROCCER estão representados por uma célula de cor cinza escuro. Para os algoritmos cujo desempenho é pior que o ROCCER, as células foram coloridas de cinza claro.

Na Tabela 5.10 são mostradas relativamente poucas diferenças estatisticamente significantes. Comparando com C4.5, ROCCER foi melhor quatro vezes e pior uma vez. Tem-se o mesmo resultado quando comparamos ROCCER com o Ripper. Comparando-se com o Slipper, o resultado é que ROCCER foi melhor cinco vezes e nenhuma vez pior. Quando se compara com C4.5 sem poda e ambas as versões do CN2, ROCCER foi duas vezes pior e nenhuma vez melhor. Uma possível explicação para esses casos pode ser a grande desproporção de exemplos entre as classes nesses conjuntos

| # | ROCCER | C45 | C45NP | CN2 | CN2OR | Ripper | Slipper | Todas |
|---|---|---|---|---|---|---|---|---|
| 1 | 98.63(1.88) | 97.76(1.51) | 98.39(1.30) | 99.26(0.81) | 99.13(0.92) | 98.72(1.38) | 99.24 (0.57) | 99.07(0.87) |
| 2 | 65.30(7.93) | 62.14(9.91) | 57.44(11.92) | 62.74(8.85) | 62.21(8.11) | 69.10(7.78) | 59.84 (6.44) | 65.38(10.63) |
| 3 | 90.31(11.56) | 50.00(0.00) | 90.06(7.75) | 90.17(6.90) | 85.15(11.38) | 61.86(25.49) | 74.78 (15.94) | 16.50(10.43) |
| 4 | 61.83(24.14) | 50.00(0.00) | 68.68(17.22) | 53.22(24.12) | 42.78(24.43) | 45.28(14.93) | 52.35 (7.44) | 62.11(23.96) |
| 5 | 72.08(6.02) | 71.43(5.89) | 67.71(4.12) | 75.25(5.38) | 70.90(4.70) | 64.02(13.62) | 71.32 (6.20) | 73.37(4.84) |
| 6 | 79.45(12.98) | 50.00(0.00) | 81.50(12.65) | 73.74(15.40) | 79.64(13.24) | 49.75(0.79) | 50.00 (2.36) | 35.62(18.93) |
| 7 | 66.41(11.54) | 55.84(6.14) | 64.33(13.58) | 59.83(9.87) | 59.28(10.13) | 57.45(3.85) | 50.40 (11.14) | 66.52(5.94) |
| 8 | 85.78(8.43) | 84.81(6.57) | 81.11(7.91) | 83.61(6.89) | 82.25(6.59) | 84.89(7.68) | 84.03 (6.36) | 90.72(6.28) |
| 9 | 94.18(4.49) | 86.09(9.97) | 90.91(6.03) | 96.23(2.97) | 92.18(7.54) | 92.06(5.94) | 93.95 (6.82) | 90.14(5.32) |
| 10 | 99.35(0.36) | 99.85(0.20) | 99.86(0.20) | 99.85(0.16) | 99.91(0.17) | 99.85(0.21) | 99.91 (0.09) | 92.67(1.60) |
| 11 | 96.08(0.52) | 95.49(1.96) | 99.33(0.46) | 99.34(0.28) | 99.44(0.63) | 97.27(1.86) | 98.82(0.44) | 92.45(1.54) |
| 12 | 98.40(1.70) | 87.85(10.43) | 97.50(3.39) | 99.14(1.19) | 98.43(2.58) | 94.95(9.94) | 99.12 (1.25) | 89.97(7.75) |
| 13 | 97.85(0.44) | 99.42(0.14) | 99.74(0.13) | 100.00(0.00) | 99.99(0.01) | 99.43(0.26) | 94.40(1.59) | 97.79(0.65) |
| 14 | 70.68(5.09) | 72.07(4.42) | 72.60(6.50) | 70.96(4.62) | 71.97(5.44) | 68.07(9.46) | 70.02 (5.97) | 70.37(5.01) |
| 15 | 89.39(2.38) | 90.15(1.70) | 91.31(1.32) | 91.48(1.45) | 91.48(0.90) | 86.83(3.94) | 89.06 (1.98) | 79.62(4.95) |
| 16 | 96.42(1.47) | 94.76(3.00) | 96.99(1.44) | 97.38(2.05) | 96.49(2.41) | 95.01(2.22) | 93.99 (3.13) | 93.37(3.05) |
| Avg | 85.13 | 77.98 | 84.84 | 84.51 | 83.2 | 79.03 | 80.08 | 75.98 |

**Tabela 5.10:** Valores da AUC estimados com validação cruzada com 10 partições em 16 conjuntos de dados da UCI descritos na Tabela 5.9, obtidos com ROCCER, C4.5, C4.5 sem poda, CN2 não ordenado, CN2 ordenado (*i.e.*, listas de decisão), Ripper, Slipper, e um classificador formado por todas as regras geradas pelo *Apriori*. Números entre parênteses indicam desvio padrão; cinza escuro significa resultados significativamente melhores com relação ao ROCCER e cinza claro indica resultados significativamente piores que ROCCER.

de dados (eles são os mais desbalanceados em nosso estudo). Para que o *Apriori* possa encontrar associações válidas para ambas as classes nesses domínios, o parâmetro de suporte do algoritmo deve ser definido com um valor muito baixo. Nesse caso, o número de regras geradas para a classe majoritária é muito grande, mas para a classe minoritária é muito pequeno. Uma possível solução para esse problema é, por exemplo, colocar um suporte mínimo diferente para cada classe. Levando em consideração todos os algoritmos de aprendizado de regras, ROCCER foi 12 vezes estatisticamente melhor que esses algoritmos e em 9 casos teve um desempenho pior. Quando comparamos com todas as regras geradas pelo *Apriori*, ROCCER foi estatisticamente melhor em 6 ocasiões, e não foi pior em nenhum dos casos. Em outras palavras, é possível afirmar que o mecanismo de seleção de regras do ROCCER é responsável por um ganho de desempenho com relação ao conjunto inicial de regras e que ROCCER tem desempenho ao menos comparável com algoritmos clássicos de indução de regras.

Os bons resultados com ambas as versões do CN2 e os valores relativamente baixos da AUC do Ripper e do Slipper podem ser explicados pela não existência de mecanismo de poda nesses dois últimos algoritmos. Resultados já reportados na literatura mostram que árvores não podadas são melhores para ordenar exemplos e, por esse motivo, produzem resultados melhores em termos da AUC (Provost & Domingos, 2003). É possível que o mesmo fenômeno também ocorra em algoritmos da família separar-para-conquistar.

Também analisamos o tamanho dos modelos gerados. Na Tabela 5.11 são apresentados o tamanho (em número médio de regras) do conjunto de regras para cada domínio e para cada algoritmo. Tamanho 0 significa que o modelo é formado apenas pela regra padrão (que geralmente classifica todos os exemplos na classe majoritária). Para a realização de testes estatísticos foi novamente utilizado o teste bicaudal de Dunnet. Dessa maneira, para cada conjunto de dados, testamos a seguinte hipótese:

$$\begin{cases} H_0: & \#Regras_{\text{ROCCER}} = \#Regras_{\text{outros algoritmos}} \\ H_1: & \#Regras_{\text{ROCCER}} \neq \#Regras_{\text{outros algoritmos}} \end{cases}$$

O teste de hipótese foi realizado com um nível de confiança de 95%. Assim como na Tabela 5.10, algoritmos cujo desempenho foram estatisticamente melhores que o ROCCER estão representados por uma célula de cor cinza escuro. Para os algoritmos cujo desempenho é pior que o ROCCER, as células foram destacadas com a cor cinza claro.

Nesse caso, a vantagem do ROCCER sobre os outros algoritmos é destacada. Afora alguma exceções, ROCCER produziu conjuntos de regras meno-

res que o C4.5 sem poda, que o CN2 em ambas as versões e que o Slipper. Em contrapartida, o Ripper produziu conjuntos significativamente menores em 7 dos 16 domínios, contra somente um significativamente menor produzido pelo ROCCER.

Analisando ambas as tabelas (Tabela 5.10 e Tabela 5.11) é possível concluir que, de maneira geral, ROCCER combina o melhor dos dois mundos: ele é capaz de gerar modelos com uma AUC comparável com os melhores algoritmos de indução de regras mas com um tamanho consideravelmente melhor que esses algoritmos.

Finalmente, na Tabela 5.12 são apresentadas estatísticas a respeito das regras individuais que compõem cada modelo, o que também monstra uma outra vantagem do ROCCER. As estatísticas apresentadas nessa tabela são ao respeito do suporte, taxa de acerto relativa ponderada (WRAcc) e razão de chances (*odds ratio*). O suporte varia de 0 a 100% e é uma medida da cobertura relativa de cada regra. A taxa de acerto relativa ponderada, também conhecida como novidade (Lavrač et al., 1999), varia entre -0.25 e 0.25 e mede a significância de uma regra em termos da diferença entre o número observado e esperado de verdadeiros positivos, como descrito na Seção 3.6. A razão entre chances varia entre 0 até $\infty$ e é uma medida da força entre a associação das duas partes da regra. Ela é definida como a razão entre as duas diagonais da matriz de contingência. Analisando essa tabela fica evidente que o ROCCER tem valores consideravelmente altos para todas essas medidas. Esse fato é um indicativo que as regras selecionadas pelo ROCCER têm isoladamente um maior significado que as regras induzidas pelos outros algoritmos.

Com respeito à complexidade, ROCCER é computacionalmente mais custoso que os outros algoritmos. No pior caso, a complexidade é da ordem de $\mathcal{O}(n^2)$, na qual $n$ é dado pelo número de regras utilizadas como entrada ao algoritmo. Entretanto, na média, o número de iterações é da ordem de $\Omega(mn)$, no qual $m$ é o número de regras selecionadas pelo algoritmo. Para a maioria dos conjuntos de dados, o tempo de execução variou entre alguns poucos segundos até 5 minutos para cada partição (em um computador Pentium 4 2.4Ghz com 512MB de RAM). Para esses conjuntos de dados, o número de regras utilizado como entrada foi de até 1.000 regras. Para alguns poucos domínios (Kr-vs-kp e Satimage) o número de regras gerados pelo *Apriori* foi muito alto (mais de 40.000). Nesses casos, o tempo de execução ficou na ordem de 1,5 horas para cada partição.

| # | ROCCER | C4.5 | C4.5NP | CN2 | CN2OR | Ripper | Slipper | Todas |
|---|---|---|---|---|---|---|---|---|
| 1 | 48.4(2.32) | 37.8(12.62) | 104.2(9.58) | 32.8(2.74) | 24.3(2.71) | 16.9(1.1) | 36.7(10.78) | 502.1(8.96) |
| 2 | 3.9(0.99) | 15(10.53) | 143.3(13.12) | 100.7(3.77) | 83.6(5.87) | 7.5(1.84) | 29.3(8.67) | 292.8(21.57) |
| 3 | 6.7(0.82) | 0(0) | 62(10.46) | 35.3(2.83) | 33.6(3.13) | 4.9(4.33) | 15.2(6.73) | 27.5(6.05) |
| 4 | 1.7(2.26) | 0(0) | 35.7(7.41) | 20.2(1.62) | 13.4(2.12) | 1.3(0.48) | 1.6(3.53) | 827.4(254.92) |
| 5 | 23.7(6.75) | 78.2(18.5) | 388.6(19.07) | 143.9(6.98) | 106.8(4.37) | 8.9(3.25) | 37.4(12.42) | 2886.1(577.3) |
| 6 | 2.4(0.52) | 0(0) | 32.1(5.99) | 21.6(1.26) | 21.9(3.18) | 0.2(0.42) | 2(3.43) | 26.7(12.1) |
| 7 | 0.8(0.42) | 5.6(5.72) | 71.2(9.39) | 75.9(3.9) | 57(5.98) | 3.5(0.97) | 11.8(13.21) | 21.2(1.4) |
| 8 | 68.2(4.42) | 13.2(4.49) | 97.4(15.04) | 42.8(2.49) | 36.1(1.79) | 7.7(1.25) | 27.2(8.73) | 1875.6(91.9) |
| 9 | 67.1(5.38) | 23.4(3.98) | 128.4(14.1) | 36.9(2.28) | 22.9(1.29) | 19.6(4.43) | 38.8(10.43) | 20855.9(2690.9) |
| 10 | 43.6(7.97) | 29.2(2.15) | 37.4(3.5) | 30.1(1.85) | 28.1(1.73) | 27.3(1.7) | 61.2(9.87) | 41790.8(565.72) |
| 11 | 78.4(3.06) | 183.8(15.22) | 700.4(31.76) | 126.3(3.56) | 120.2(2.7) | 71(6.58) | 115.2(6.16) | 649.1(20.55) |
| 12 | 8.5(0.53) | 4(0) | 35.4(4.2) | 18.7(0.67) | 15.3(0.82) | 10.9(3.21) | 20.7(5.48) | 39.1(4.72) |
| 13 | 18.9(1.66) | 114.6(3.1) | 227.1(3.57) | 112.4(2.27) | 17.6(0.52) | 44.1(7.32) | 57.6(2.37) | 100.0(4.29) |
| 14 | 4(0.82) | 49.4(20.27) | 347.4(10) | 169.7(10.08) | 168.4(7.9) | 8.7(2.21) | 30.5(10.82) | 10.7(3.62) |
| 15 | 143.9(51.28) | 531.1(84.63) | 1767.2(111.93) | 158.8(6.09) | 199.8(6.99) | 32(4.37) | 52.3(13.9) | 4451.2(712.9) |
| 16 | 48.4(4.48) | 89.5(12.12) | 188.9(9.42) | 49.8(3.29) | 41.2(2.82) | 23.7(4.06) | 75.7(12.11) | 431.4(53.85) |
| Avg | 35.54 | 73.43 | 272.92 | 73.49 | 61.89 | 18.01 | 38.33 | 4674.23 |

**Tabela 5.11:** Número médio de regras obtidos nos experimentos reportados na Tabela 5.10. Números entre parênteses indicam desvio padrão; cinza escuro significa resultados significativamente melhores com relação ao ROCCER e cinza claro indica resultados significativamente piores que ROCCER.

| | Suporte (%) | WRAcc | Razão de chances |
|---|---|---|---|
| ROCCER | 13.67 (13.89) | 0.0355 (0.018) | 154.02 (337.33) |
| C4.5 | 3.73 (6.01) | 0.0094 (0.013) | 44.55 (77.04) |
| C4.5NP | 1.19 (1.06) | 0.0030 (0.003) | 31.90 (56.68) |
| CN2 | 3.90 (2.52) | 0.0110 (0.009) | 98.73 (138.64) |
| CN2OR | 3.10 (2.18) | 0.0085 (0.007) | 95.07 (192.94) |
| Ripper | 5.96 (5.34) | 0.0184 (0.012) | 74.08 (103.88) |
| Slipper | 1.92 (1.58) | 0.0060 (0.006) | 33.86 (50.41) |
| All | 8.07 (5.06) | 0.0114 (0.014) | 67.39 (111.72) |

**Tabela 5.12:** Suporte, precisão relativa ponderada e razão de chances médias de todas as regras induzidas

## 5.3. Considerações finais

As duas principais contribuições descritas neste capítulo são as duas novas abordagens para a geração de regras. Na primeira delas, é possível extrair exceções a partir de regras gerais de classificação. Essa abordagem permite que haja uma melhor intepretação semântica das regras geradas.

A segunda abordagem baseia-se no uso da análise ROC para selecionar regras a partir de um conjunto maior de regras. As regras selecionadas formam uma lista de decisão. Em geral, o algoritmo tem um desempenho comparável, medido em termos da AUC, a vários algoritmos de aprendizado de regras descritos na literatura. Um aspecto no qual o ROCCER se destaca tem a ver com o número de regras selecionadas que, na maioria das vezes, é sistematicamente inferior ao número de regras induzidas pelos outros algoritmos.

Capítulo 6

# Classes desbalanceadas: entendendo o problema

Neste capítulo é apresentada uma introdução ao problema de classes desbalanceadas. Na Seção 6.1 é feita uma introdução ao problema, no qual algumas das classes são geralmente muito mais representadas no conjunto de exemplos que outras. Na Seção 6.2 são apresentados experimentos realizados com conjuntos de dados artificiais com o intuito de ganhar um melhor entendimento a respeito do problema de classes desbalanceadas e confirmar nossa hipótese de que a sobreposição de exemplos entre as classes é um fator complicante desse problema. Finalmente, na Seção 6.3 são apresentadas as considerações finais deste capítulo.

## 6.1. Classes desbalanceadas

Muitos aspectos podem influenciar o desempenho de um modelo de classificação criado por um sistema de aprendizado supervisionado. Um desses aspectos está correlacionado com a diferença entre o número de exemplos pertencentes a cada uma das classes. Quando essa diferença é grande, os sistemas de aprendizado podem encontrar dificuldades em induzir o conceito relacionado à classe minoritária. Nessas condições, modelos de clas-

sificação que são otimizados em relação à precisão têm tendência de criar modelos triviais, que quase sempre predizem a classe majoritária.

Entretanto, em muitos dos problemas reais, uma grande desproporção no número de casos pertencentes a cada uma das classes é comum. Por exemplo, na detecção de fraudes em chamadas telefônicas (Fawcett & Provost, 1997) e transações realizadas com cartões de crédito (Stolfo et al., 1997), o número de transações legítimas é muito maior que o de transações fraudulentas. Na modelagem de risco de seguros (Pednault et al., 2000), apenas uma pequena porcentagem dos segurados reclama suas apólices em um dado período. Outros exemplos de domínios com um desbalanceamento intrínseco entre as classes podem ser encontrados na literatura. Além disso, em muitas aplicações, não se sabe qual é a proporção exata de exemplos pertencentes a cada classe ou se essa proporção pode variar no tempo.

Muitos sistemas de aprendizado, nessas circunstâncias, não estão preparados para induzir modelos de classificação que possam predizer acertadamente as classes minoritárias. Freqüentemente, esses modelos têm uma boa precisão na classificação da classe majoritária, mas a precisão para a classe minoritária não é aceitável. O problema é ainda maior quando o custo associado a uma classificação errônea para a classe minoritária é muito maior que o custo de uma classificação errônea para a classe majoritária. Infelizmente, esta é a norma e não a exceção para a maioria das aplicações com conjuntos de dados desbalanceados, pois geralmente a classe minoritária é a de maior interesse.

Vários pesquisadores têm analisado o problema de aprender a partir de conjuntos de dados com classes desbalanceadas (Pazzani et al., 1994; Ling & Li, 1998; Kubat & Matwin, 1997; Fawcett & Provost, 1997; Kubat et al., 1998; Weiss, 2004; Weiss & Provost, 2003; Chawla, 2005; Han et al., 2005; Tang & Liu, 2005). Além disso, ocorreram dois whorkshops internacionais relacionados a esse tema, um deles patrocinado pela Associação Americana de Inteligência Artificial — American Association for Artificial Intelligence (AAAI)— (Japkowicz, 2001) e o outro acontecido em conjunto com a vigésima Conferência Internacional de Aprendizado de Máquina — International Conference on Machine Learning (ICML'03) —(Chawla et al., 2003). Também, a revista do grupo de estudos em mineração de dados da ACM, SIGKDD Explorations, dedicou uma edição especial ao tema (Chawla et al., 2004).

Dentre os diversos métodos propostos nesses trabalhos, três abordagens principais têm sido utilizadas com maior freqüência. São elas:

**Atribuição de custos de classificação incorreta.** Em muitos domínios de aplicação, classificar incorretamente exemplos da classe minoritária é mais custoso do que classificar incorretamente exemplos da classe majoritária. Para esses domínios, é possível utilizar sistemas de aprendizado sensíveis ao custo de classificação. Esses sistemas objetivam minimizar o custo total ao invés da taxa de erro de classificação. A principal restrição ao uso desses sistemas é que o custo de classificação incorreta de cada classe deve ser previamente conhecido e deve ser um valor constante;

**Remoção de exemplos das classes majoritárias.** Essa abordagem é geralmente chamada de *under-sampling*. Uma forma bastante direta de solucionar o problema de classes desbalanceadas é balancear artificialmente a distribuição das classes no conjunto de exemplos. Os métodos de *under-sampling* visam balancear o conjunto de dados por meio da remoção de exemplos das classes majoritárias;

**Inclusão de exemplos das classes minoritárias.** Métodos dessa categoria são geralmente conhecidos como métodos de *over-sampling*. Os métodos de *over-sampling* são similares aos métodos de *under-sampling*. Entretanto, esses métodos visam balancear a distribuição das classes por meio da replicação de exemplos da classe minoritária.

A atribuição de custos e a inclusão ou remoção de exemplos estão fortemente ligados. Uma forma de aprender com conjuntos com classes desbalanceadas é treinar um sistema sensível ao custo, com o custo da classe minoritária maior do que o da classe majoritária. Uma outra forma de fazer com que um sistema de aprendizado se torne sensível ao custo é alterar intencionalmente a distribuição das classes no conjunto.

Por exemplo, para sistemas que não são capazes de lidar com custos, Breiman et al. (1984) propõem um método simples e geral para torná-los sensível ao custo para um problema de duas classes. Esse método baseia-se em modificar a distribuição das classes no conjunto de treinamento de forma a aumentar o número de exemplos da classe mais custosa. Suponha que a classe positiva é cinco vezes mais custosa que a classe negativa. Se o número de exemplos positivos for artificialmente aumentado por um fator de cinco, então o sistema de aprendizado, visando reduzir os custos de classificação, irá induzir um modelo que tenda a evitar cometer erros na classe positiva, uma vez que qualquer erro nessa classe é penalizado cinco vezes mais. Domingos (1999) expande essa idéia e apresenta um método geral para tornar qualquer sistema de aprendizado sensível ao custo. Esse método possui a vantagem de ser aplicável a problemas que possuem qual-

quer número de classes. Elkan (2001a) demonstra um teorema que permite encontrar a proporção de exemplos positivos e negativos de forma a fazer classificações sensíveis ao custo ótimas para um problema de duas classes.

Em suma, a maioria dos métodos que tratam conjuntos de dados com classes desbalanceadas visam melhorar o desempenho da classe minoritária por meio do balanceamento de classes do conjunto de dados ou atribuição de custos. Como essas duas abordagens estão ligadas, incrementando o número de exemplos da classe minoritária torna essa classe mais custosa, e pode-se esperar que ela será melhor classificada. Paralelamente, atribuir um custo maior aos exemplos da classe minoritária faz com que os sistemas tendam a se focar nesses exemplos para melhorar a sua função objetivo (que passa a ser o custo total e não mais a taxa de erro de classificação) e espera-se que a classe minoritária seja melhor classificada.

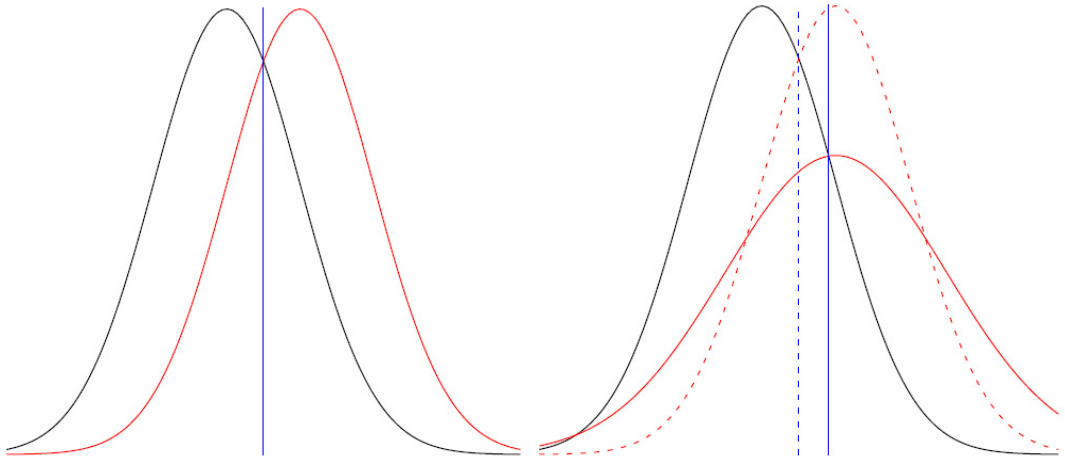
### 6.1.1 O desbalanceamento é sempre um problema?

Muitos trabalhos com classes desbalanceadas apontam esse problema como o maior responsável pelo baixo desempenho de algoritmos de aprendizado em domínios nos quais algumas classes são pouco representadas. Entretanto, algoritmos de aprendizado têm um desempenho surpreendentemente bom, com taxa de acerto e AUC próximos de 100%, mesmo em domínios altamente desbalanceados (Laurikkala, 2001; Batista et al., 2004). Esse fato é um bom indicativo de que a proporção de exemplos entre as classes não é sempre a responsável pelo mau desempenho dos algoritmos de aprendizado em domínios com um desbalanceamento natural das classes. Em outras palavras, mesmo que em alguns domínios com classes desbalanceadas algoritmos de aprendizado tenham um desempenho ruim, o desbalanceamento não é, necessariamente, o culpado por esse mau desempenho.

Para ilustrar o problema, considere um problema de decisão muito simples mostrado na Figura 6.1. O problema está relacionado a construir um modelo de classificação para um problema de apenas um atributo e duas classes: positiva e negativa. As probabilidades condicionais para ambas as classes são dadas por funções Gaussianas unidimensionais com variância um. O valor médio de atributo para a classe negativa está uma unidade deslocado para a direita.

No problema representado na Figura 6.1(a), o objetivo é construir um modelo de classificação bayesiano assumindo um perfeito conhecimento com respeito às distribuições de probabilidade. Nesse caso, é possível construir o modelo de classificação ótimo de Bayes. A linha vertical representa a divisão ótima entre as classes, segundo o critério de Bayes. Nessas condi-

ções, o modelo ótimo não é alterado, não importando o quão desbalanceado é o conjunto de exemplos. Em contrapartida, na Figura 6.1(b) é apresentado o mesmo problema, só que agora não se conhece de antemão as distribuições *a priori*. Em outras palavras, as distribuições devem ser estimadas a partir dos dados. Caso haja uma grande desproporção entre as classes, é provável que os algoritmos de aprendizado produzam estimativas ruins para as classes com poucos exemplos. Na Figura 6.1(b) em particular, a variância é superestimada em 1,5 (linha contínua) ao invés da variância real 1 (linha tracejada). Ou seja conhecendo-se de antemão as probabilidades condicionais (uma restrição dificilmente aplicável em problemas do mundo real) de maneira que seja possível construir o modelo ideal de Bayes, a distribuição de exemplos entre as classes não representa um problema. Em contrapartida, quando se utiliza somente os dados disponíveis para estimar os parâmetros, e esses dados não são suficientes para gerar estimativas confiáveis, classes desbalanceadas podem ser um problema na indução de modelos para a classificação de exemplos.



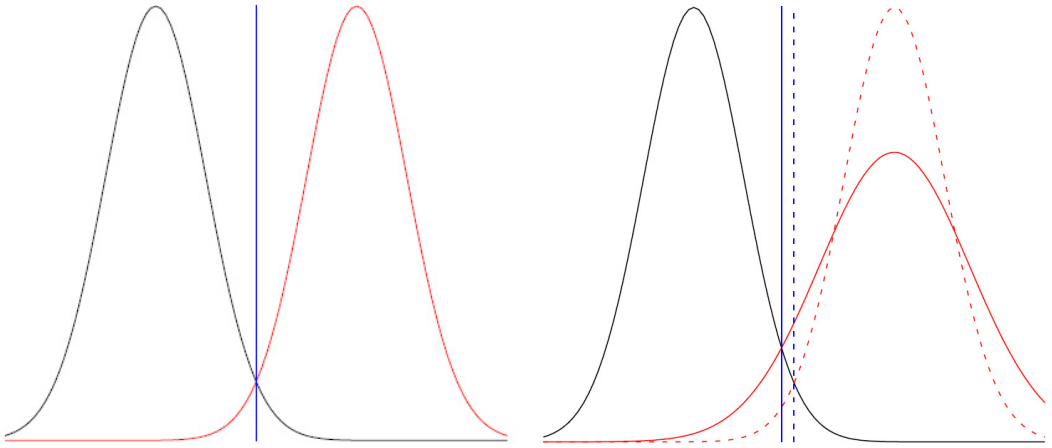
(a) O algoritmo tem completa informação sobre o domínio.

(b) Somente os dados disponíveis são utilizados para a indução do modelo.

**Figura 6.1:** Um problema com grande sobreposição de classes.

Consideremos agora um problema ligeiramente diferente, mostrado na Figura 6.2. Novamente, o problema é composto por um único atributo e duas classes que seguem uma distribuição Gaussiana, mas a média dos valores desse atributo para a classe negativa está agora quatro (ao invés de uma) unidades à direita da classe positiva. Assim como na Figura 6.1(a), na Figura 6.2(a) é representado o cenário em que completo conhecimento sobre as distribuições de probabilidades é assumido. Já na Figura 6.2(b) é representado o cenário em que o algoritmo de aprendizado deve estimar esses valores somente com os dados disponíveis. Pelas mesmas razões descritas anteriormente, quando é assumido perfeito conhecimento a respeito

das distribuições, o modelo ótimo de Bayes não é afetado pela distribuição de exemplos entre as classes. Entretanto, no caso em que não se conhece *a priori* essas distribuições, o modelo final é afetado. Note que, uma vez que as médias dos valores do atributo para as duas classes estão mais distantes, as classes estão mais separadas. Isso é equivalente a dizer que existe uma menor sobreposição entre as classes. Observe que, por esse motivo, o efeito do desbalanceamento entre as classes é menor do que no caso com uma maior sobreposição de classes.



(a) O algoritmo tem completa informação sobre o domínio.

(b) Somente os dados disponíveis são utilizados para a indução do modelo.

**Figura 6.2:** Um problema com pequena sobreposição de classes.

Note também que mesmo que a distância entre as linhas verticais pontilhada e contínua seja semelhante, a área entre elas e a curva que representa a curva de distribuição de probabilidade é maior na Figura 6.1(b) do que na Figura 6.2(b). Isso é equivalente a dizer que o número de exemplos classificados incorretamente no primeiro caso é maior do que o número de exemplos classificados incorretamente no segundo caso. Esse é um indicativo que o desbalanceamento entre as classes não é o único responsável pelo baixo desempenho em problemas desbalanceados, mas também o grau de sobreposição entre as classes. Na Seção 6.2 é apresentada uma série de experimentos com conjuntos de dados artificiais com o intuito de confirmar essa hipótese.

### 6.1.2 Métodos para balancear artificialmente conjuntos de dados.

Como mencionado previamente, além de atribuir custos diferentes às classes, uma das maneiras mais diretas de lidar com classes desbalanceadas é alterar a distribuição dessas classes de forma a tornar o conjunto de

dados mais balanceado. Nesta seção são descritos os métodos para balancear artificialmente os conjuntos de dados utilizados neste trabalho[1].

Existem dois métodos básicos para balancear a distribuição das classes: remover exemplos das classes mais populosas e inserir exemplos nas classes menos populosas. Em suas versões mais simples, essa adição/remoção é feita de maneira aleatória. Nesses casos, os métodos são geralmente chamados de *over-sampling* aleatório e *under-sampling* aleatório:

***Over-sampling* aleatório** é um método não heurístico que replica aleatoriamente exemplos da classe minoritária. Em nossa implementação, essa replicação é feita sem reposição.

***Under-sampling* aleatório** também é um método não heurístico que elimina aleatoriamente exemplos da classe majoritária.

Por serem aleatórios, ambos os métodos possuem limitações conhecidas. *Under-sampling* aleatório pode eliminar dados potencialmente úteis, e *over-sampling* aleatório pode aumentar as chances de ocorrer superajustamento aos dados, uma vez que cópias exatas dos exemplos pertencentes à classe minoritária são duplicados. Dessa maneira, em um modelo simbólico, por exemplo, pode-se construir regras que são aparentemente gerais, mas que na verdade cobrem um único exemplo replicado. Alguns trabalhos recentes têm tentado superar as limitações existentes desses métodos. Esses métodos geralmente utilizam heurísticas para selecionar os exemplos a serem acrescentados/removidos, cujo principal objetivo é tentar minimizar a quantidade de dados potencialmente úteis descartados.

Algumas dessas heurísticas exploram os exemplos em uma das seguintes categorias: **ruído** são casos que estão do lado errado da borda de decisão; **redundantes** são casos que podem ser representados por outros casos que estão presentes no conjunto de treinamento; **próximos à borda** são casos que estão próximos da borda de decisão; **seguros** são casos que não são ruído, não estão excessivamente próximos à borda de decisão e, também, não estão muito distantes dela. Ruído é indesejável em qualquer condição de aprendizado. Exemplos redundantes não acrescentam muita informação ao sistema de aprendizado e casos próximos à borda são pouco confiáveis pois não se tem certeza sobre o verdadeiro local da fronteira entre uma classe e outra. Por exemplo, uma pequena quantidade de ruído pode mover esses exemplos para o lado errado da fronteira de decisão.

Os métodos utilizados neste trabalho são descritos a seguir:

[1] Os métodos descritos nesta seção foram implementados na biblioteca DOL (Batista & Monard, 2003b; Batista, 2003), que é parte do sistema para descoberta de conhecimento DISCOVER (Prati, 2003; Prati et al., 2003b), que vem sendo desenvolvido em nosso laboratório de pesquisa.

**Ligações Tomek** Casos próximos à borda e ruído podem ser identificados por meio das ligações Tomek, e removidos do conjunto de dados. Uma ligação Tomek pode ser definida da seguinte maneira:

> *Sejam $e_i$ e $e_j$ dois exemplos de classes diferentes. Seja d uma função de distância entre exemplos. Um par de exemplos $(e_i, e_j)$ constitui uma ligação Tomek se não existe um exemplo $e_k$, tal que a distância $d(e_k, e_i) < d(e_i, e_j)$ ou $d(e_k, e_j) < d(e_i, e_j)$.*

Se dois exemplos $(e_i, e_j)$ formam uma ligação Tomek, então, ou $e_i$ e $e_j$ são exemplos próximos à borda de decisão, ou um desses exemplos é possivelmente ruído. No uso de ligações Tomek para o balanceamento de conjuntos de dados, apenas os exemplos da classe majoritária que possuem ligações Tomek são removidos.

**Regra do vizinho mais próximo condensada** Parte dos casos redundantes pode ser removida por meio da identificação de um subconjunto consistente. Um subconjunto $\hat{E} \subset E$ é consistente com $E$ se utilizando o algoritmo do 1-vizinhos-mais-próximo (1-NN) ele classifica corretamente os casos em $E$ (Hart, 1968). Essa abordagem é geralmente conhecida como regra do vizinho mais próximo condensada— CNN (*Condensed Nearest Neighbor Rule*) (Hart, 1968). Um algoritmo para a geração de subconjunto consistente consiste nos seguintes passos: Primeiramente é selecionado aleatoriamente um exemplo da classe majoritária e todos os exemplos da classe minoritária, que são inseridos em $\hat{E}$. A seguir, usa-se o algoritmo 1-NN sobre o conjunto $\hat{E}$ para classificar os exemplos em $E$. Todo exemplo incorretamente classificado de $E$ é movido para $\hat{E}$. A idéia por detrás desse algoritmo é remover exemplos que estão muito distantes da fronteira de decisão, uma vez que esses exemplos são geralmente considerados menos importantes no processo de aprendizagem. Também é importante notar que esse algoritmo não encontra o conjunto consistente ótimo a partir de $E$.

**Seleção unilateral** (Kubat & Matwin, 1997) é um método que consiste da aplicação do algoritmo para a identificação de ligações Tomek seguido da aplicação do CNN. Essa abordagem é conhecida como seleção unilateral. Ligações Tomek são utilizadas para identificar exemplos da classe majoritária que se sobrepõem à classe minoritária e CNN é utilizado para remover exemplos muito distantes da fronteira de decisão. O restante dos exemplos, *i.e.*, exemplos da classe majoritária "seguros" e todos os exemplos da classe minoritária, são utilizados para a aprendizagem.

**CNN + ligações Tomek** em (Batista et al., 2004) propomos a inversão dos passos da seleção unilateral. Como a identificação de ligações Tomek é computacionalmente custosa, ela é computacionalmente mais efetiva se aplicada em um conjunto reduzido.

**Regra de limpeza da vizinhança** o método da limpeza do vizinho (Laurikkala, 2001) — NCL (*Neighborhood Cleaning Rule*) — utiliza a regra do vizinho mais próximo ENN (*Edited Nearest Neighbor Rule*) de Wilson (1972) para remover exemplos da classe majoritária. ENN remove exemplos cuja classe difere da classe por, pelo menos, 2 dos seus 3 vizinhos mais próximos.

**SMOTE** Chawla et al. (2002) propõem um método que não replica os exemplos de treinamento. Nesse trabalho, o método de *over-sampling* cria novos exemplos da classe minoritária por meio da interpolação de diversos exemplos dessa classe que se encontram próximos. Esse método é chamado de SMOTE (*Synthetic Minority Over-sampling Technique*).

**SMOTE + ligações Tomek** mesmo que os métodos de *over-sampling* possam balancear a distribuição de exemplos entre as classes, outros problemas presentes em conjuntos de dados desbalanceados não são resolvidos. Frequentemente, os exemplos que compõem a classe minoritária não estão bem agrupados e pode haver uma grande sobreposição entre as classes. Com o objetivo de melhorar a definição dos agrupamentos de dados no conjunto de treinamento, em (Batista et al., 2004) propomos a aplicação do método para a identificação de ligações Tomek após a aplicação do método SMOTE. Diferentemente da aplicação de ligações Tomek como método de *under-sampling*, nesse caso (como o conjunto de dados foi previamente balanceado com exemplos "sintéticos") removemos exemplos de ambas as classes.

**SMOTE + ENN** também propomos a aplicação do método ENN após a aplicação do método SMOTE. A motivação é similar a do método SMOTE + ligações Tomek. Entretanto, ENN tende a remover mais exemplos do que as ligações Tomek, promovendo uma maior limpeza no conjunto de dados. Similarmente ao método SMOTE + ligações Tomek, ENN é utilizado para remover exemplos de ambas as classes.

## 6.2. Experimentos com conjuntos de dados artificiais

Apesar de extremamente utilizados para a validação empírica de estudos em aprendizado de máquina, a realização de estudos com conjuntos de dados naturais muitas vezes não são apropriados para se fornecer *insights* a respeito das causas do porque uma abordagem é melhor que a outra. Isso se deve ao fato que não se conhece o processo que gera tais dados. *Insights* são geralmente obtidos pela execução de experimentos em conjuntos de dados artificiais, especialmente desenvolvidos para se testar uma hipótese específica. A importância de se utilizar conjuntos artificiais está relacionada ao fato de que eles permitem que se variem sistematicamente as características do domínio de interesse, tais como o número de atributos relevantes ou irrelevantes, quantidade de ruído, número de classes, desbalanceamento de classes, complexidade do conceito, e outros. Dessa maneira, eles permitem que os pesquisadores testem hipóteses a respeito do desempenho de cada método. Nesta seção é descrita uma série de experimentos realizados com conjuntos de dados artificiais, com o objetivo de procurar um melhor entendimento do problema de desbalanceamento entre as classes.

### 6.2.1 Configuração experimental

Para realizar nossos experimentos foram gerados 10 domínios artificiais. Esses domínios são compostos por dois agrupamentos: um representando a classe majoritária e o outro representando a minoritária. Os exemplos gerados no experimento têm dois parâmetros controlados. O primeiro é a distância entre os centros dos agrupamentos de cada uma das classes e o segundo é a proporção de exemplos em cada uma das classes. A distância entre os centros dos agrupamentos nos permite controlar o "grau de dificuldade" de classificar corretamente as duas classes. A proporção de exemplos em cada classe nos permite analisar o desempenho de algoritmos de aprendizado em diferentes condições de proporções de exemplos para cada uma das classes.

Cada domínio é composto por 5 atributos. Os atributos são independentes entre si, e cada atributo é proveniente de uma distribuição Gaussiana com variância 1. Além disso, cada domínio tem duas classes, designadas pelo rótulo genérico **positivo** (classe minoritária) e **negativo** (classe majoritária). Para o primeiro domínio, o centro de cada Gaussiana é o mesmo para ambas as classes. Para os domínios restantes, é incrementado o centro (média) de cada atributo em uma unidade, até o máximo de nove. Para cada domínio foram gerados catorze conjuntos de dados. Cada conjunto tem

10.000 exemplos com diferentes proporções de exemplos em cada classe, variando-a de 1% até 45% os exemplos pertencentes à classe positiva. O restante dos exemplos pertencem à classe negativa. As proporções utilizadas foram as seguintes: 1%, 2.5%, 5%, 7.5% 10%, 12.5%, 15%, 17.5%, 20%, 25%, 30%, 35%, 40% e 45%. Também incluímos um conjunto de dados de controle com a mesma distribuição de exemplos em ambas as classes.

Mesmo que a complexidade dos domínios gerados seja aparentemente simples (geramos conjuntos de dados com somente 5 atributos, duas classes e cada classe agrupada em somente um grupo), essa situação é freqüentemente enfrentada por algoritmos de aprendizado supervisionado, uma vez que a maioria desses algoritmos utiliza abordagens recursivas (como as estratégias dividir-para-conquistar ou separar-para-conquistar). Além disso, a distribuição Gaussiana pode ser utilizada como aproximação de várias outras distribuições estatísticas.

A utilização de funções Gaussianas também nos permite calcular facilmente valores teóricos da AUC para o modelo ótimo de Bayes. A AUC pode ser calculada utilizando-se a Equação 6.1 (Marzban, 2004), na qual $\Phi(.)$ é a distribuição normal cumulativa padrão, $\delta$ é a distância euclidiana entre os centros das duas distribuições, e $\phi_{pos}$ e $\phi_{neg}$ são, respectivamente, o desvio padrão do centróide das classes positivas e negativas.

$$AUC = \Phi\left(\frac{\delta}{\sqrt{\phi_{pos} + \phi_{neg}}}\right) \tag{6.1}$$

Para executar os experimentos utilizamos o algoritmo C4.5 (Quinlan, 1993) para a indução de árvores de decisão. Dois motivos nos levaram a escolher o C4.5. Primeiramente, a indução de árvores é um dos métodos mais utilizados na construção de modelos de classificação. Além disso, C4.5 é muito utilizado na avaliação de algoritmos de aprendizado em domínios com classes desbalanceadas. Modificamos a árvore induzida pelo C4.5 para predizer probabilidades ao invés de predizer somente a classe, como proposto por (Provost & Domingos, 2003). Os resultados foram avaliados utilizando-se a AUC. Todos os experimentos foram realizados utilizando-se validação cruzada com 10 partições.

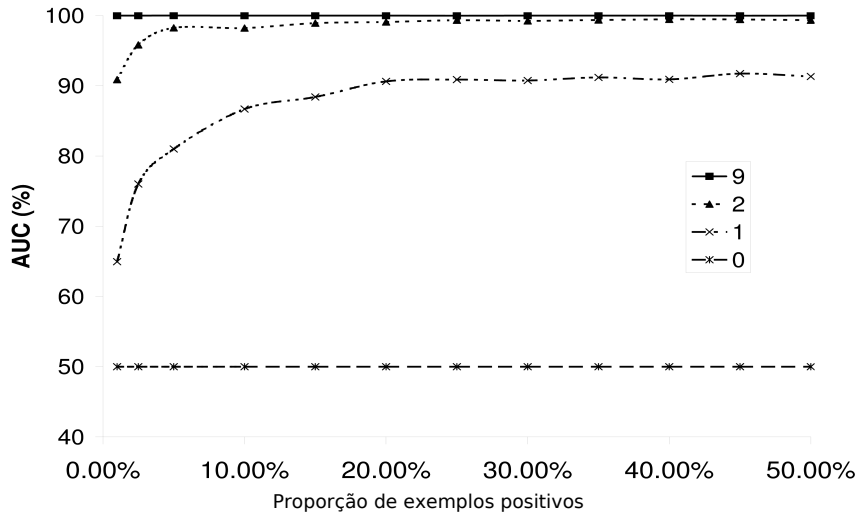
### 6.2.2 Sobreposição de classes e desbalanceamento

Deve ser observado que, apesar do baixo desempenho em alguns conjuntos de dados desbalanceados, para alguns domínios em que esse problema ocorre algoritmos de aprendizado têm um desempenho surpreendentemente bom (Laurikkala, 2001). Esse fato nos levou a conjectura de que o desba-

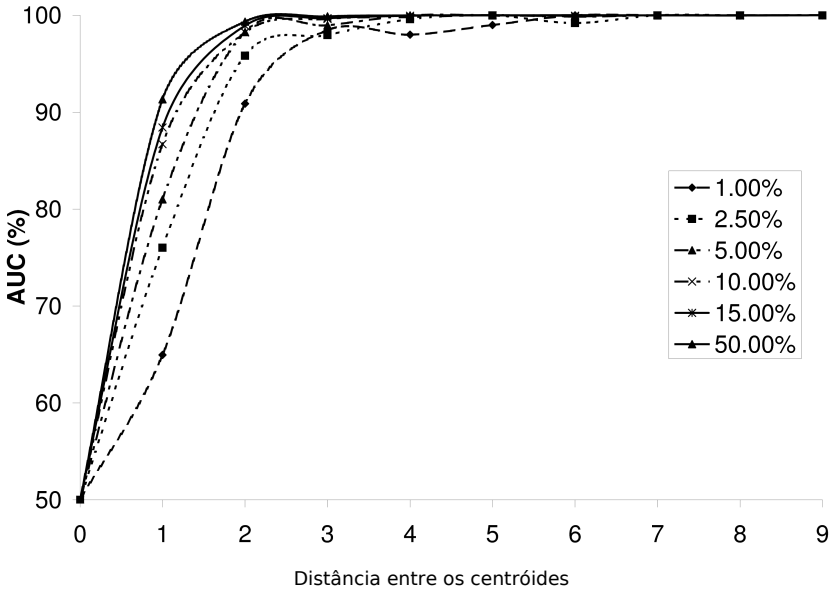
lanceamento entre as classes não é sempre um fator determinante para o baixo desempenho de algoritmos de aprendizado em domínios em que esse problema ocorre. Em (Prati et al., 2004a), como descrito na Seção 6.1.1 consideramos a hipótese de que o grau de sobreposição entre as classes é um fator que colabora para um baixo desempenho em domínios com classes desbalanceadas.

Na Figura 6.3 estão sumarizados os principais resultados de nosso trabalho. Para uma melhor visualização, omitimos algumas proporções e distâncias entre os centros das classes. Entretanto, as linhas omitidas são muito similares àquelas cujas distâncias entre os centros dos agrupamentos de cada classe estão separados 9 unidades entre si e a proporção de exemplos é de 50% em cada classe. Na Figura 6.3(a) estão representadas a porcentagem de exemplos pertencentes à classe positiva no conjunto de dados *versus* a AUC dos modelos induzidos pelo C4.5 para diferentes distâncias entre os centróides da classe positiva e da classe negativa. Considere a curva em que a classe positiva está duas unidades deslocada da classe negativa. Observe que os modelos induzidos para essa distância têm um ótimo desempenho, com AUC maior que 90%, mesmo que a proporção de exemplos da classe positiva seja de somente 1% dos exemplos.

Na Figura 6.3(b) estão representadas a variação do centróide da classe positiva *versus* a AUC dos modelos induzidos pelo C4.5 para diferentes proporções de exemplos entre as classes. Nesse gráfico pode ser observado que a maior degradação no desempenho dos modelos ocorre principalmente quando a diferença entre os centróides das classes positiva e negativa é de uma unidade. Nesse caso, a degradação é significativamente alta para conjuntos de dados muito desbalanceados, mas diminui conforme a distância entre os centróides aumenta. A diferença entre o desempenho do algoritmo para domínios com distância entre os centróides maior que três unidades é estatisticamente insignificante, independentemente de quantos exemplos pertencem à classe positiva. Esses resultados estão de acordo com a nossa hipótese de que o problema de desbalanceamento entre as classes é maior quando existe uma alta sobreposição entre as classes. Um experimento similar, também com conjuntos de dados artificiais, mas gerados de maneira diferente do reportado neste trabalho (Japkowicz, 2003), também chegou à conclusão de que o desbalanceamento entre as classes não é sempre o responsável pela degradação de desempenho em algoritmos de aprendizado. Japkowicz (2003) conclui que o problema depende tanto da "complexidade do conceito", na qual complexidade do conceito corresponde ao número de subgrupos em que as classes estão divididas, e do tamanho do conjunto de




(a) Variação na proporção de exemplos positivos *versus* AUC.




(b) Variação no centróide da classe positiva *versus* AUC.

**Figura 6.3:** Resultados experimentais do C4.5 aplicado a conjuntos de dados com diversos graus de sobreposição e desbalanceamento entre as classes.

dados. Nossa conclusão é similar a essa, mas em nosso caso "complexidade do conceito" é medida como o grau de sobreposição entre as classes.

### 6.2.3 Sobreposição de classes e métodos de balanceamento

Verificada empiricamente a nosso hipótese de que a sobreposição de exemplos entre as classes é um fator que potencializa o problema de classes desbalanceadas, em (Batista et al., 2005) analisamos o comportamento

de métodos que artificialmente promovem o balanceamento do conjunto de treinamento em conjuntos de dados com sobreposição entre as classes.

Como os conjuntos de dados cujas distâncias entre os centróides da classe positiva e negativa são maiores do que 4 unidades não apresentam diferenças significativas em termos da AUC, nesta seção nos focalizamos em conjuntos de dados com distância entre os centróides menor que 3. Além disso, geramos novos domínios com as mesmas características dos descritos na Seção 6.2.1, mas com distância entre os centróides variando em $0,5$. Dessa maneira, nesse experimento analisamos 7 domínios, com as seguintes distâncias entre os centróides das classes: $0$, $0,5$, $1$, $1,5$, $2$, $2,5$ e $3$. Os resultados dos valores teóricos da AUC para essas distâncias são mostrados na Tabela 6.1. Como estamos interessados na interação entre conjuntos de dados com um alto grau de desbalanceamento e sobreposição de classes, também restringimos a análise para domínios com até 20% de exemplos na classe positiva, e comparamos os resultados com o conjunto de dados naturalmente balanceado.

| $\delta$ | 0 | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 |
|---|---|---|---|---|---|---|---|
| AUC | 50.0 % | 78.54 % | 94.31 % | 99.11 % | 99.92 % | 99.99 % | 99.99 % |

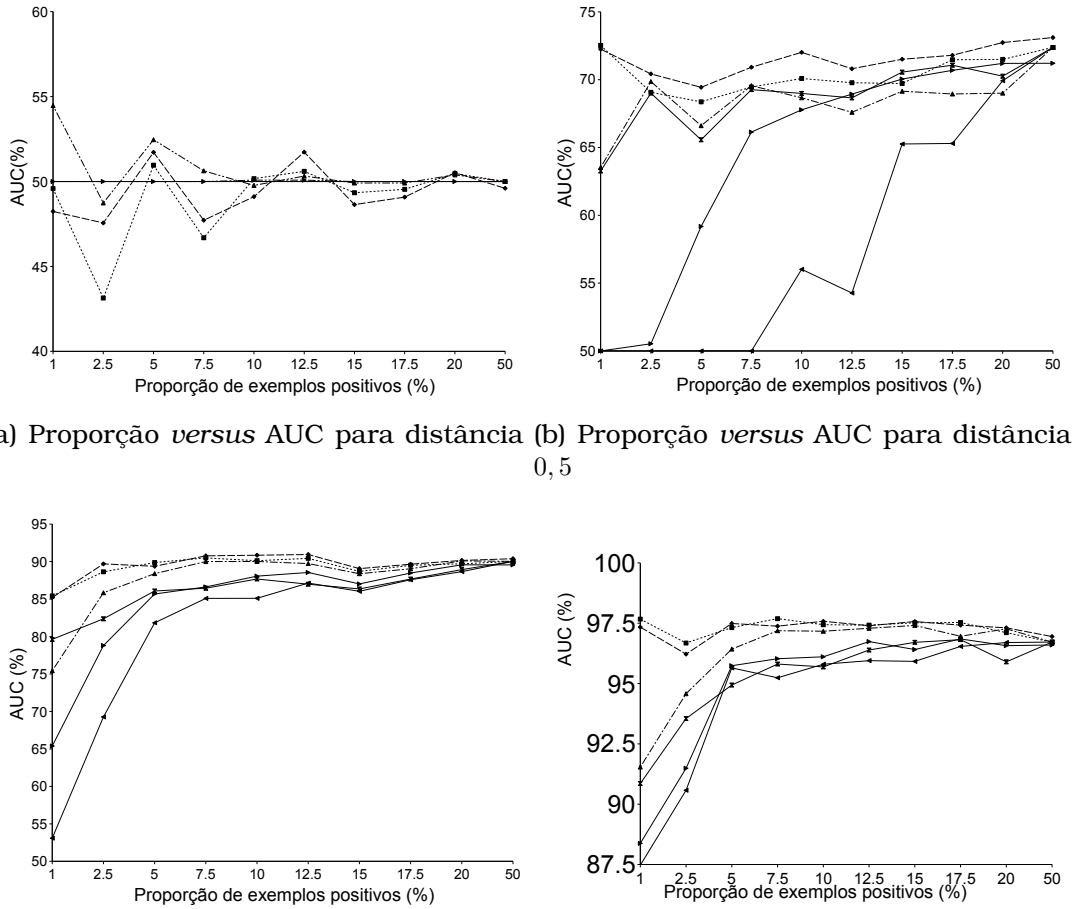
**Tabela 6.1:** Valores teóricos da AUC para os conjuntos de dados artificiais gerados

Nesse experimento, utilizamos os métodos de balanceamento SMOTE, SMOTE + ENN, NCL, *over-sampling* aleatório e *under-sampling* aleatório — Seção 6.1.2. Nossa implementação do SMOTE, *over-sampling* aleatório e *under-sampling* aleatório têm parâmetros internos que permitem especificar a proporção final de exemplos entre as classes obtida depois da aplicação desses métodos. Decidimos adicionar/remover exemplos até que a proporção de exemplos entre as classes seja igualada. Essa decisão foi motivada por resultados experimentais apresentados em (Weiss & Provost, 2003) que afirmam que, na média, quando se usa AUC como medida de avaliação, a melhor distribuição de exemplos para induzir árvores de decisão tende a ser a balanceada.

Para uma melhor visualização dos resultados, eles são apresentados graficamente. Na Figura 6.4 são apresentados os resultados dos experimentos para distâncias entre os centróides $0$, $0,5$, $1$ e $1,5$. Note que, para uma melhor visualização dos resultados, a escala dos eixos não é a mesma em todos os gráficos.

Esses gráficos mostram, para cada distância entre os centróides, a AUC média (calculada utilizando-se validação cruzada com 10 partições) *versus* a proporção de exemplos positivos nos conjuntos de dados de treinamento. A distância 0 (Figura 6.4(a)) foi introduzida para fins de comparação. Como

esperado, os valores da AUC para essa distância oscilam (devido a variações aleatórias) em torno do desempenho aleatório (AUC = 50%). Nos experimentos, a maior influência do desbalanceamento entre as classes ocorre quando a distância entre os centros é de 0.5. Nesse caso, o valor teórico da AUC é de 78.54%, mas com 15% dos exemplos positivos no conjunto de treinamento a AUC está abaixo dos 65%.




(a) Proporção *versus* AUC para distância 0

(b) Proporção *versus* AUC para distância $0,5$

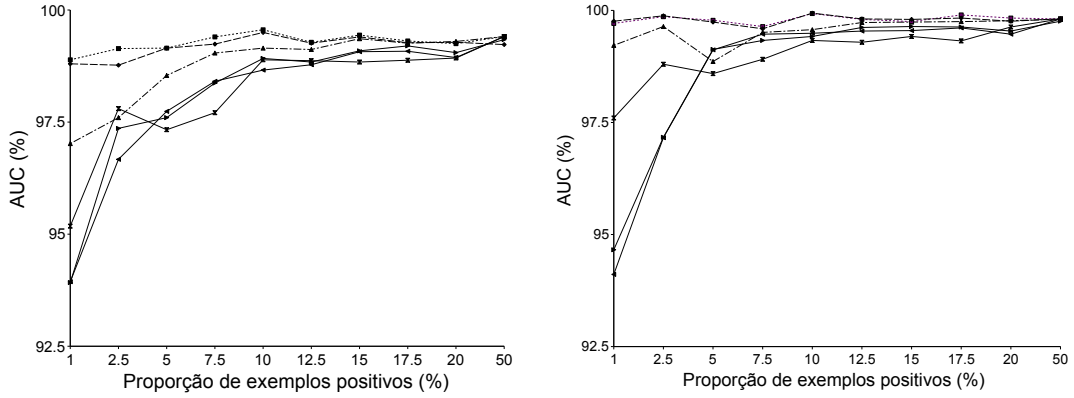
(c) Proporção *versus* AUC para distância 1

(d) Proporção *versus* AUC para distância $1,5$

**Figura 6.4:** Resultados experimentais para as distâncias entre os centróides 0, 0.5, 1 e 1.5.

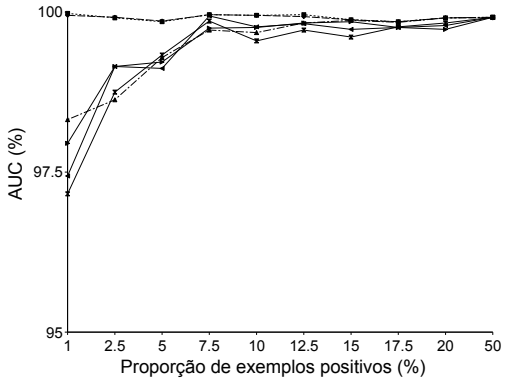
Na maioria dos casos, os métodos de balanceamento foram capazes de melhorar o desempenho dos algoritmos de aprendizado, medidos em termos da AUC. Como pode ser observado, *under-sampling* aleatório e NCL apresentam uma pequena melhora sobre o conjunto de dados original. Entretanto, essa melhora é menor do que a obtida com os métodos de *over-sampling*. Os métodos que adicionam exemplos à classe positiva geralmente produziram os melhores resultados, com os métodos baseados no SMOTE obtendo um desempenho quase que constante para todas as distribuições de exemplos.

O método SMOTE + ENN apresentou melhores resultados que os outros métodos na maioria das distribuições de exemplos. Esse fato pode ser explicado pela aplicação do método de limpeza de dados, que tendem a ser mais eficientes em regiões mais sobrepostas. O método de limpeza ENN é menos efetivo conforme a distância aumenta, uma vez que existem menores regiões sobrepostas a serem limpas quando a distância entre os grupos de exemplos de cada classe aumentam. Desse maneira, quanto mais distantes os centróides das classes, mais os métodos SMOTE e SMOTE+ENN tendem a apresentar resultados similares. Para a distância $1,5$, a maioria dos métodos apresentou bons resultados, com os valores da AUC superiores a 90%. O método *over-sampling* aleatório teve uma AUC em torno de 97%. Entretanto, os métodos baseados no SMOTE apresentaram resultados quase que constantes para todos as distribuições de exemplos, mesmo quando os conjuntos de exemplos eram extremamente desbalanceados.




(a) Proporção *versus* AUC para distância $2$

(b) Proporção *versus* AUC para distância $2,5$




(c) Proporção *versus* AUC para distância $3$

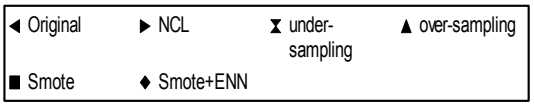


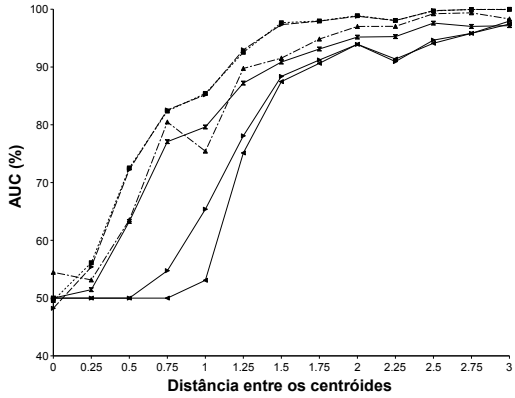

**Figura 6.5:** Resultados experimentais para as distâncias entre os centróides $2$, $2,5$ e $3$.

Na Figura 6.5 são apresentados os resultados para os conjuntos de dados cuja distância entre os centróides está entre $2$, $2,5$ e $3$. Para essas distâncias, os métodos que incrementam o número de exemplos da classe positiva também obtiveram os melhores resultados, especialmente para os conjuntos de dados muito desbalanceados. Os resultados do SMOTE e SMOTE+ENN são ligeiramente melhores que *over-sampling* aleatório. Entretanto, a limpeza de dados feita pelo ENN é pouco efetiva e os resultados de SMOTE e SMOTE+ENN são muito semelhantes. Observe que os métodos baseados no SMOTE obtiveram uma AUC quase que constante e com valores próximos a 100% para todas as distribuições de exemplos entre as classes. É interessante notar o baixo desempenho do método *over-sampling* aleatório para conjuntos de dados altamente desbalanceados. Como o *over-sampling* aleatório replica identicamente os exemplos da classe positiva, esse fato pode ser devido a um superajustamento do algoritmo a esses exemplos replicados.

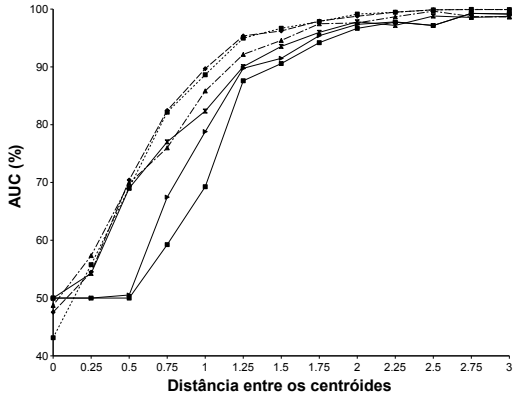
Para concluir as análises desta seção, é importante notar que estamos geralmente interessados em métodos que produzem os melhores resultados em conjuntos de dados muito desbalanceados. Para analisar esse problema em maiores detalhes, na Figura 6.6 são mostrados os resultados obtidos variando-se a distância entre o centróide de cada classe para as distribuições de exemplos entre as classes mais desbalanceadas: 1%, 2,5% e 5% dos exemplos positivos. Esses gráficos mostram claramente que os métodos que aumentam o conjunto de exemplos positivos, e os métodos baseados no SMOTE em particular, apresentam os resultados mais significativos.

## 6.3. Considerações finais

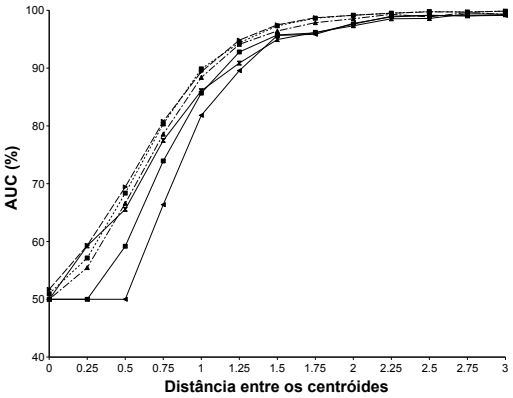
Neste capítulo foi apresentada uma introdução ao problema de aprendizado com classes desbalanceadas. Esse problema ocorre quando o número de exemplos disponíveis para algumas das classes é muito pequeno. Com base na hipótese de que a sobreposição de exemplos entre as classes é um fator complicante ao aprendizado com classes desbalanceadas três novos métodos de balanceamento artificial de conjuntos de dados foram propostos: CNN + Tomek, SMOTE + Tomek e SMOTE + ENN. Também realizamos uma série de experimentos para confirmar a nossa hipótese que o problema de classes desbalanceadas se agrava na presença de uma grande sobreposição de exemplos entre as classes. Nessa série de experimentos utilizamos conjuntos de dados gerados artificialmente para verificar a nossa hipótese. Também verificamos que, nessas condições, métodos de *over-sampling* têm melhores resultados pois lidam mais diretamente com o problema. No pró-




(a) Variação do centro *versus* AUC para a proporção de 1% de exemplos positivos




(b) Variação do centro *versus* AUC para a proporção de 2,5% de exemplos positivos




(c) Variação do centro *versus* AUC para a proporção de 5% de exemplos positivos

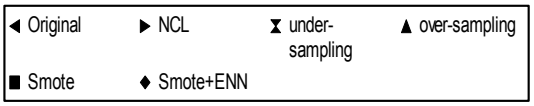



**Figura 6.6:** Resultados experimentais para as proporções 1%, 2.5% e 5%.

ximo capítulo, o problema de classes desbalanceadas é investigado em maiores detalhes com conjuntos de dados reais provenientes da UCI, além da investigação de outros problemas relacionados à proporção de exemplos entre as classes.

Capítulo 7

# Aprendizado e a proporção de exemplos entre as classes

Neste capítulo são investigados outros problemas relacionados ao aprendizado com diferentes proporções de exemplos entre as classes. Na Seção 7.1 são apresentados experimentos em conjuntos de dados naturais provenientes do repositório de dados da UCI para testar a eficiência de métodos de balancemento artificial de conjuntos de dados desbalanceados, três deles propostos nesse trabalho. Na Seção 7.2 são brevemente apresentados trabalhos relacionados ao problema de classes desbalanceadas e pequenos disjuntos, e no processo de rotulação de exemplos em aprendizado semi-supervisionado multi-visão utilizando Co-training. Finalmente, na Seção 7.3 são apresentadas as considerações finais deste capítulo.

## 7.1. Experimentos com conjuntos de dados naturais

Além dos experimentos com conjuntos de dados artificiais, descritos no capítulo anterior, também realizamos uma série de experimentos com conjuntos de dados naturais provenientes do repositório da UCI (Newman et al., 1998). Esses experimentos estão reportados a seguir.

### 7.1.1 Avaliação experimental de vários métodos de balanceamento de classes

Nesta Seção são descritos os experimentos reportados em (Batista et al., 2004), com o intuito de comparar vários métodos de balanceamento de conjuntos de dados naturais e averiguar a eficácia desses métodos em problemas reais com desbalanceamento entre as classes. Para realizar essa comparação, escolhemos treze conjuntos de dados com diferentes graus de desbalanceamento do repositório de dados da UCI. Na Tabela 7.1 são sumarizadas as características dos conjuntos de dados utilizados no estudo. Os conjuntos de dados estão ordenados em ordem crescente de desbalanceamento. Como os conjuntos Letter e Splice têm um número de exemplos similar nas classes minoritárias, criamos dois conjuntos de dados para cada um: Letter-a e Letter-vogais, nos quais as classes minoritárias são "a" e "todas as vogais", respectivamente, e Splice-ie e Splice-ei, nos quais as classes minoritárias são "ie" e "ei", respectivamente.

| # | Conj. dados | $n_{ex}$ | $n_{atr}$ | Classe maj.% |
|---|---|---|---|---|
| 1 | Pima | 768 | 8 (8,0) | 65,23% |
| 2 | German | 1000 | 20 (7,13) | 70,00% |
| 3 | Post-operative | 90 | 8 (1,7) | 73,33% |
| 4 | Haberman | 306 | 3 (3,0) | 73,53% |
| 5 | Splice-ie | 3176 | 60 (0,60) | 75,91% |
| 6 | Splice-ei | 3176 | 60 (0,60) | 76,01% |
| 7 | Vehicle | 846 | 18 (18,0) | 76,48% |
| 8 | Letter-vowel | 20000 | 16 (16,0) | 80,61% |
| 9 | New-thyroid | 215 | 5 (5,0) | 83,72% |
| 10 | E.Coli | 336 | 7 (7,0) | 89,58% |
| 11 | Satimage | 6435 | 36 (36,0) | 90,27% |
| 12 | Flag | 194 | 28 (10,18) | 91,24% |
| 13 | Glass | 214 | 9 (9,0) | 92,06% |
| 14 | Letter-a | 20000 | 16 (16,0) | 96,05% |
| 15 | Nursery | 12960 | 8 (8,0) | 97,45% |

**Tabela 7.1:** Descrição dos conjuntos de dados utilizados em experimentos com classes desbalanceadas. $n_{ex}$ é o número de exemplos. $n_{atr}$ é o número de atributos. Valores entre pararêntesis indicam o número de atribudos discretos e contínuos, respectivamente. Classe maj.% é porcentagem de exemplos na classe majoritáira.

Novamente os experimentos foram executados com o algoritmo C4.5. Os experimentos foram executados utilizando-se validação cruzada com 10 partições. Os resultados são mostrados na Tabela 7.2. Nessa tabela, para cada conjunto de dados, é mostrada a AUC da aplicação do C4.5 no conjunto de dados original, bem como dos diversos métodos para o balanceamento dos conjuntos de dados, na ordem: *under-sampling* aleatório,

CNN, CNN + Tomek, Tomek, Tomek + CNN, NCL, *over-sampling* aleatório, SMOTE, SMOTE + ENN e SMOTE + Tomek. Além disso, para cada conjunto de dados, são apresentados os resultados da árvore construída utilizando os parâmetros padrão do C4.5 (poda com 25% de confiança) e sem poda.

Para facilitar a análise dos resultados apresentados na Tabela 7.2, derivamos vários gráficos baseados nesses resultados. O primeiro gráfico é mostrado na Figura 7.1. Nessa figura estão representados o valor médio da AUC *versus* a proporção de exemplos negativos/positivos para os conjuntos de dados originais. Nessa figura fica evidente que nem sempre o desbalanceamento nos conjuntos de dados leva a uma degradação de desempenho, uma vez que conjuntos extremamente desbalanceados, tais como o Letter-a e Nursery, obtiveram AUC muito próximas de 100%.

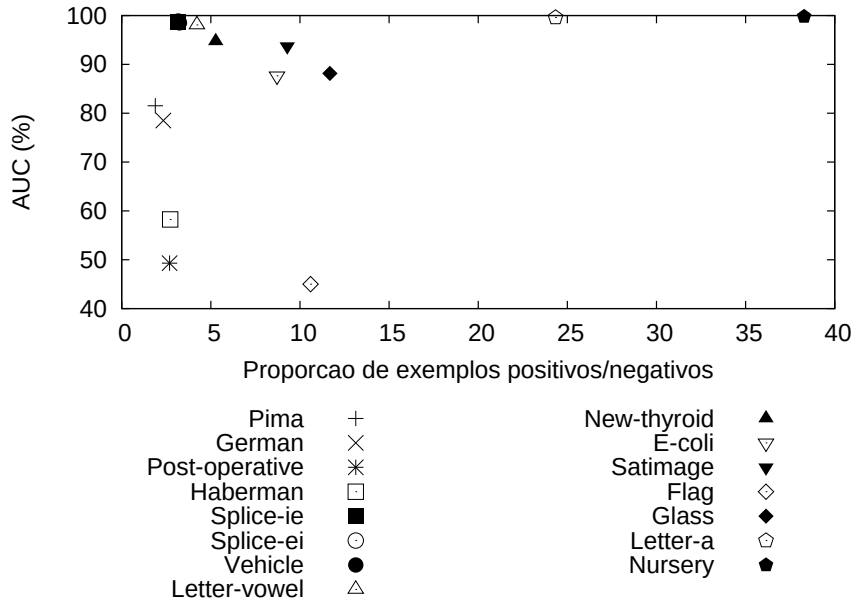



**Figura 7.1:** Proporção de exemplos positivos/negativos *versus* AUC.

Esses resultados obtidos nos experimentos com os conjuntos de dados da UCI são compatíveis com os experimentos em conjuntos de dados artificiais, reportados na Seção 6.2.2. Em outras palavras, esses resultados corroboram com a hipótese de que o desbalanceamento entra as classes não é sempre um problema para algoritmos de aprendizado.

Quanto ao tamanho do conjunto de treinamento na presença de conjuntos de dados desbalanceados e a degradação de desempenho (reportado em Japkowicz (2003) como um fator que potencializa o problema do desbalanceamento entre as classes), nossos experimentos mostram que o problema pode estar relacionado com a baixa presença de exemplos da classe minoritária no conjunto de treinamento. Conjuntos de dados pequenos aliados a um desbalanceamento entre as classes implicam em um número reduzido de exemplos na classe minoritária, e esse número reduzido de exem-

| # | poda | original | *Under* aleatório | CNN | CNN Tomek | Tomek | Tomek CNN | NCL | *Over* aleatório | SMOTE | SMOTE ENN | SMOTE Tomek |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | sim | 81.53(5.11) | 81.17(3.87) | 79.60(6.22) | 80.30(3.86) | 82.56(5.11) | 77.89(5.37) | 81.61(4.48) | 85.32(4.17) | 85.49(5.17) | 84.46(5.84) | 83.66(4.77) |
| | não | 82.33(5.70) | 81.49(4.29) | 80.08(5.82) | 81.71(3.69) | 83.11(4.65) | 79.23(4.81) | 82.55(3.53) | 86.03(4.14) | 85.97(5.82) | 85.56(6.02) | 83.64(5.35) |
| 2 | sim | 78.49(7.75) | 78.58(7.99) | 80.01(8.06) | 77.60(9.05) | 77.78(8.36) | 78.26(7.53) | 77.83(9.33) | 83.74(6.73) | 79.97(7.16) | 80.29(7.69) | 79.32(8.51) |
| | não | 85.67(4.37) | 85.12(4.89) | 82.27(6.93) | 83.82(4.39) | 84.75(4.67) | 83.09(6.26) | 84.70(6.24) | 85.03(4.91) | 84.19(5.54) | 84.40(6.39) | 82.76(5.93) |
| 3 | sim | 49.29(2.26) | 49.11(14.07) | 49.20(8.91) | 49.02(11.34) | 46.16(5.89) | 46.31(18.77) | 42.34(28.12) | 68.79(23.93) | 55.66(24.66) | 41.80(16.59) | 59.83(33.91) |
| | não | 78.23(15.03) | 55.52(24.47) | 65.69(21.64) | 75.79(16.86) | 66.45(23.29) | 64.44(20.88) | 45.62(32.71) | 71.33(23.43) | 68.19(26.62) | 47.99(16.61) | 59.48(34.91) |
| 4 | sim | 58.25(12.26) | 66.07(10.26) | 58.36(10.26) | 55.73(14.31) | 64.46(10.95) | 62.70(11.50) | 68.01(13.99) | 71.81(13.42) | 72.23(9.82) | 75.73(6.55) | 76.38(5.51) |
| | não | 67.91(13.76) | 68.40(10.17) | 58.36(10.26) | 55.73(14.31) | 69.59(13.30) | 62.03(11.82) | 69.29(14.13) | 73.58(14.22) | 75.45(11.02) | 78.41(7.11) | 77.01(5.10) |
| 5 | sim | 98.76(0.56) | 97.46(1.10) | 98.39(0.64) | 97.55(0.46) | 98.69(0.51) | 97.37(0.84) | 98.38(0.57) | 98.89(0.47) | 98.46(0.87) | 98.26(0.51) | 97.97(0.74) |
| | não | 99.30(0.30) | 98.80(0.40) | 99.17(0.36) | 98.82(0.32) | 99.18(0.43) | 98.93(0.30) | 99.15(0.36) | 99.09(0.27) | 99.19(0.28) | 99.13(0.31) | 98.88(0.34) |
| 6 | sim | 98.77(0.46) | 98.74(0.46) | 98.78(0.46) | 98.85(0.42) | 98.78(0.46) | 98.83(0.45) | 98.77(0.47) | 98.80(0.44) | 98.92(0.44) | 98.87(0.44) | 98.85(0.60) |
| | não | 99.47(0.61) | 99.25(0.48) | 99.27(0.77) | 99.47(0.27) | 99.44(0.60) | 99.33(0.66) | 99.40(0.66) | 99.52(0.60) | 99.52(0.26) | 99.51(0.32) | 99.49(0.16) |
| 7 | sim | 98.49(0.84) | 97.25(1.95) | 98.62(0.67) | 98.34(1.32) | 98.26(0.90) | 98.79(0.67) | 97.94(1.05) | 99.14(0.73) | 98.96(0.98) | 98.96(0.98) | 97.92(1.09) |
| | não | 98.45(0.90) | 97.80(0.94) | 98.64(0.63) | 98.42(1.02) | 98.41(0.90) | 98.71(0.97) | 98.17(1.12) | 99.13(0.75) | 99.04(0.85) | 99.04(0.85) | 98.22(0.90) |
| 8 | sim | 98.07(0.63) | 97.69(0.43) | 98.03(0.37) | 97.97(0.46) | 98.18(0.53) | 97.66(0.30) | 98.17(0.30) | 98.80(0.32) | 98.90(0.20) | 98.90(0.20) | 98.94(0.22) |
| | não | 98.81(0.33) | 98.26(0.28) | 98.49(0.31) | 98.39(0.22) | 98.90(0.18) | 98.27(0.19) | 98.81(0.17) | 98.84(0.27) | 99.15(0.17) | 99.14(0.17) | 99.19(0.15) |
| 9 | sim | 94.73(9.24) | 94.87(5.00) | 94.79(10.14) | 94.54(10.10) | 94.73(9.24) | 92.72(10.55) | 93.44(9.74) | 98.39(2.91) | 98.91(1.84) | 98.91(1.84) | 99.22(1.72) |
| | não | 94.98(9.38) | 94.87(5.00) | 94.79(10.14) | 94.54(10.10) | 94.98(9.38) | 92.72(10.55) | 93.69(9.90) | 98.89(2.68) | 98.91(1.84) | 98.91(1.84) | 99.22(1.72) |
| 10 | sim | 87.64(15.75) | 88.75(12.45) | 80.32(19.96) | 80.34(19.85) | 91.57(7.81) | 83.97(21.27) | 91.73(8.00) | 93.24(6.72) | 95.49(4.30) | 95.98(4.21) | 95.29(3.79) |
| | não | 92.50(7.71) | 88.64(12.46) | 81.13(20.00) | 81.95(19.90) | 94.03(5.56) | 83.76(21.17) | 92.04(8.15) | 93.55(6.89) | 95.49(4.30) | 95.98(4.21) | 95.29(3.79) |
| 11 | sim | 93.73(1.91) | 92.34(1.27) | 92.25(1.45) | 92.73(1.38) | 94.21(1.76) | 92.85(1.19) | 94.42(1.53) | 95.34(1.25) | 95.43(1.03) | 95.43(1.03) | 95.67(1.18) |
| | não | 94.82(1.18) | 92.86(1.29) | 92.35(1.35) | 92.90(1.38) | 95.11(1.29) | 92.84(1.22) | 95.06(1.27) | 95.52(1.12) | 95.69(1.28) | 95.69(1.28) | 96.06(1.20) |
| 12 | sim | 45.00(15.81) | 71.13(28.95) | 49.12(21.57) | 75.85(30.26) | 45.00(15.81) | 45.00(15.81) | 44.47(15.71) | 79.91(28.72) | 73.62(30.16) | 79.30(28.68) | 79.32(28.83) |
| | não | 76.65(27.34) | 78.35(29.98) | 78.90(28.63) | 75.64(29.37) | 78.59(28.75) | 81.73(29.51) | 76.13(27.80) | 79.78(28.98) | 73.87(30.34) | 82.06(29.52) | 78.56(28.79) |
| 13 | sim | 88.16(12.28) | 82.44(8.99) | 58.44(13.15) | 72.69(14.07) | 87.15(16.47) | 72.16(16.84) | 91.67(12.76) | 92.20(12.11) | 91.40(8.24) | 91.40(8.24) | 92.90(7.30) |
| | não | 88.16(12.28) | 80.47(13.25) | 64.31(14.21) | 75.44(11.61) | 87.00(16.75) | 78.76(12.52) | 91.67(12.76) | 92.07(12.09) | 91.27(8.38) | 91.27(8.38) | 93.40(7.61) |
| 14 | sim | 99.61(0.40) | 99.35(0.48) | 99.60(0.37) | 99.61(0.37) | 99.61(0.40) | 99.66(0.46) | 99.60(0.40) | 99.77(0.30) | 99.91(0.12) | 99.91(0.12) | 99.91(0.12) |
| | não | 99.67(0.37) | 99.46(0.42) | 99.66(0.37) | 99.65(0.38) | 99.67(0.37) | 99.67(0.45) | 99.67(0.37) | 99.78(0.29) | 99.92(0.12) | 99.92(0.12) | 99.91(0.14) |
| 15 | sim | 99.79(0.11) | 97.52(0.82) | 99.55(0.21) | 98.77(0.35) | 99.80(0.08) | 99.47(0.19) | 99.79(0.12) | 99.99(0.01) | 99.21(0.55) | 99.27(0.36) | 97.80(1.07) |
| | não | 99.96(0.05) | 98.76(0.22) | 99.84(0.13) | 99.57(0.21) | 99.89(0.08) | 99.83(0.08) | 99.89(0.09) | 99.99(0.01) | 99.75(0.34) | 99.53(0.31) | 99.20(0.51) |

**Tabela 7.2:** AUC para os conjuntos de dados da Tabela 7.1 originais e depois de aplicados vário métodos para o balanceamento artificial desses conjuntos. Células coloridas em cinza clara indicam o melhor valor de AUC obtido para esses conjuntos.

plos pode ser insuficiente para se aprender o conceito relacionado àquela classe. Para conjuntos maiores, o efeito desse fator complicante tende a ser menor, uma vez que mesmo que o conjunto de dados seja muito desbalanceado, a classe minoritária pode estar melhor representada. Essa tendência é confirmada no gráfico apresentado na Figura 7.2, que mostra como a AUC é afetada pelo número (absoluto) de exemplos positivos no conjunto de treinamento.

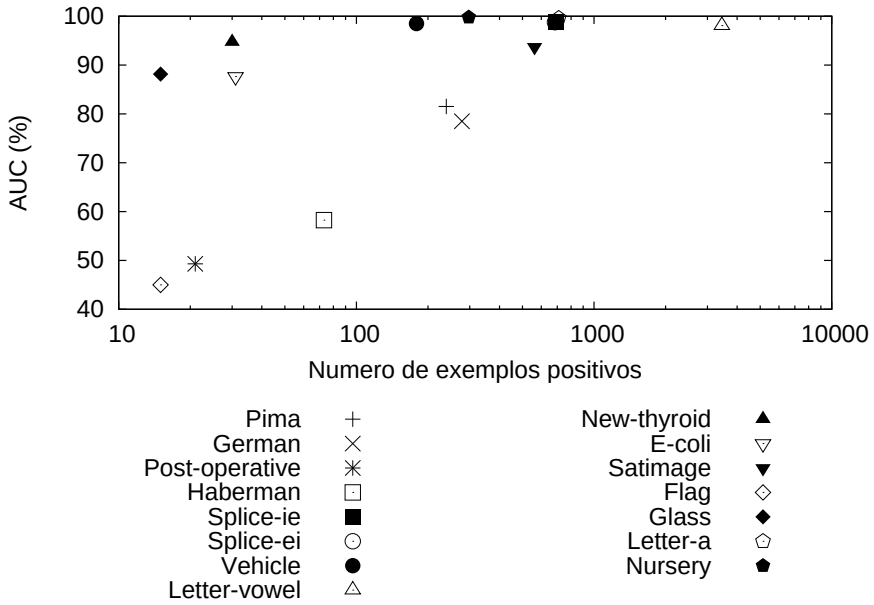



**Figura 7.2:** Número absoluto de exemplos positivos *versus* AUC.

Também analisamos o relacionamento da poda da árvore de decisão e o desbalanceamento entre as classes. Alguns trabalhos apontam que a poda pode ser útil em alguns casos (Chawla, 2003). No entanto, outros trabalhos (Provost & Domingos, 2003; Zadrozny & Elkan, 2001; Bauer & Kohavi, 1999) concluem que quando se quer estimativas de probabilidade mais confiáveis, ou quando os custos de classificação ou a distribuição de exemplos entre as classes é desconhecida, a poda deve ser evitada. Uma das razões para se evitar a poda é que vários dos métodos de poda, incluindo o usado pelo C4.5, têm como finalidade minimizar o erro. Esse fato pode prejudicar as classes minoritárias, uma vez que reduzindo o erro da classe majoritária (cortando ramos que predizem a classe minoritária, por exemplo) tem-se um maior impacto na taxa de erro global. Além do mais, mesmo que, via de regra, os sistemas que induzem árvore de decisão realizem poda, a questão de se podar ou não uma árvore na qual o conjunto de treinamento foi modificado para se tentar balancear artificialmente o conjunto de dados parece ser um problema em aberto. Um argumento contra a poda, nesses casos, é que se ela for realizada nessas condições, o sistema de aprendizado poderia podar com uma falsa suposição, *i.e.*, que a distribuição de exemplos

no conjunto de treinamento é a mesma na qual ele será aplicado (Weiss & Provost, 2003).

Na Figura 7.3 é mostrada uma comparação do efeito da poda nas árvores de decisão induzidas a partir dos conjuntos de dados originais e artificialmente balanceados. A linha diagonal $x = y$ representa AUC idênticas para a árvore podada e não podada. Pontos acima dessa linha representam o fato que árvores não podadas obtiveram resultados melhores e pontos abaixo dessa linha indicam o oposto. Observando a Figura 7.3, podemos concluir que a poda raramente melhora o desempenho (medido em termos da AUC) da árvore induzida, tanto para conjuntos de dados com distribuições originais quanto para conjuntos de dados artificialmente balanceados.

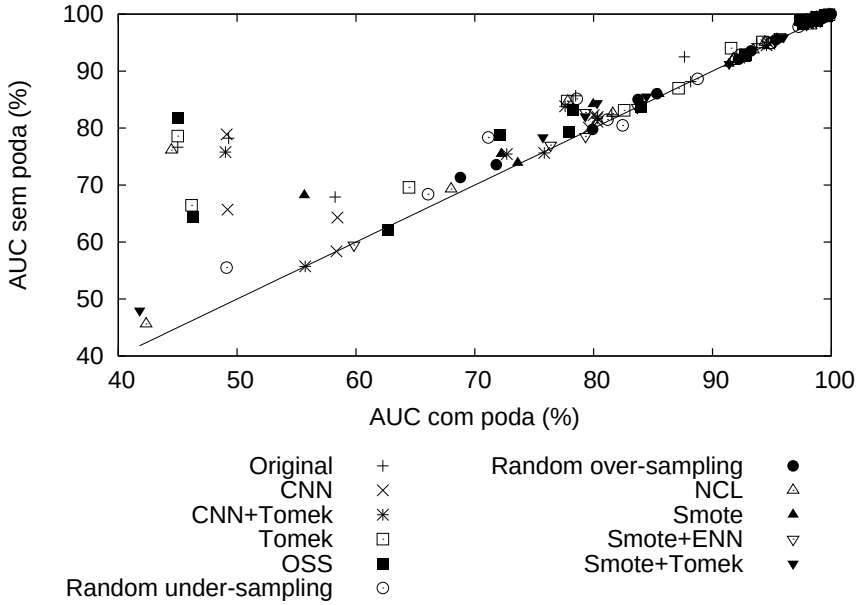



**Figura 7.3:** AUC das árvores podadas *versus* árvores sem poda para os conjuntos de dados originais e artificialmente balanceados.

Para facilitar a análise dos métodos de balanceamento, as células que contêm os valores de AUC mais altos para cada conjunto de dados e para ambas as árvores podadas e não podadas na Tabela 7.2, estão coloridas de cinza claro. Note que a maioria dos melhores resultados se concentra nos métodos que aumentam o conjunto de dados, com alguns poucos para a distribuição original de exemplos. Para uma melhor visualização dos resultados, nas Tabelas 7.3 e 7.4 é apresentado]a uma ordenação dos métodos com relação à AUC para árvores podadas e não podadas, respectivamente. As células que contêm os resultados para os métodos de *over-sampling* estão destacadas com uma cor cinza claro. As células que contêm os resultados para os conjuntos de dados originais estão destacadas com cinza escuro. Note que, em geral, os métodos de *under-sampling* aparecem depois dos métodos de *over-sampling*. Para realizar inferências a respeito dos métodos de

balanceamento, aplicamos o teste de múltiplas comparações com o melhor de Hsu (1996) — Multiple Comparison with the Best (MCB) — para testar, para cada conjunto de dados, se existem diferenças significativas entre a AUC máxima e as demais. Dessa maneira, testamos para cada conjunto de dados a seguinte hipótese:

$$\begin{cases} H_0: & \text{Método com máxima AUC} = \text{outros métodos} \\ H_1: & \text{Método com máxima AUC<outros métodos} \end{cases}$$

o teste foi realizado com um grau de confiança de 95%. Os resultados da aplicação do teste estão sumarizados nas Tabelas 7.3 e 7.4. Os métodos nos quais foi possível rejeitar $H_0$, *i.e.*, métodos que obtiveram um AUC significativamente menor que o melhor método, com um nível de confiança de 95%, estão marcados com o símbolo $\nabla$.

Em contrapartida, os métodos de *over-sampling* em geral, e o *over-sampling* aleatório em particular, estão bem ordenados entre os outros métodos. Esses resultados estão em discordância com alguns outros trabalhos publicados na literatura. Drummond & Holte (2003), por exemplo, reportam que quando é usado C4.5 com os seus parâmetros padrão, *over-sampling* é aparentemente ineficiente, muitas vezes produzindo poucas alterações no desempenho em resposta a mudanças nos custos de classificação ou distribuição de exemplos entre as classes. Além disso, eles notam que após a aplicação de *over-sampling*, as árvores tendem a ser menos podadas do que quando se aplica *under-sampling*, e que a generalização é menor (tamanho da árvore é maior) quando se aplica *over-sampling*. Nossos experimentos mostram que, na maioria dos casos, *over-sampling* aleatório aparentemente não está se super-ajustando aos dados, mesmo em árvores não podadas, como pode ser notado levando-se em consideração os altos valores da AUC obtidos por esse método. Além disso, os métodos de *under-sampling* não apresentaram bons desempenhos, mesmo quando se utiliza heurísticas para remover casos da classe majoritária.

Outros resultados contraditórios incluem Domingos (1999), que reporta que em problemas binários, o algoritmo C4.5 Rules[1] produz modelos com menores custos de classificação utilizando-se *under-sampling* do que aqueles obtidos utilizando-se *over-sampling*. Ling & Li (1998) comparam *over-sampling* e *under-sampling* aleatório para uma versão do C4.5 com aplicação de *boosting* e reportam que *under-sampling* leva a melhores índices *lift*[2], mas com *over-sampling* extremo, *i.e.*, se a classe minoritária é muito incre-

[1] Versão modificada do C4.5 que extrai regras a partir da árvore originalmente induzida.

[2] *Lift* é uma media que indica o quanto uma predição é mais correta do que um modelo trivial que sempre classifica aquela classe. Ela é definida como a taxa de acerto dividida pela porcentagem de exemplos da classe positiva.

| # | 1º | 2º | 3º | 4º | 5º | 6º | 7º | 8º | 9º | 10º | 11º |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Smote | *Over* aleatório | Smote + Tomek | Smote + ENN | Tomek | NCL | Original | *Under* aleatório | CNN + Tomek | CNN∇ | OSS∇ |
| 2 | *Over* aleatório | Smote + Tomek | Smote + ENN | Smote | *Under* aleatório | CNN | CNN + Tomek∇ | OSS∇ | Original∇ | Tomek∇ | NCL∇ |
| 3 | *Over* aleatório | Smote + ENN | Smote | Original | CNN | *Under* aleatório | CNN + Tomek | OSS∇ | Tomek∇ | NCL∇ | Smote + Tomek∇ |
| 4 | Smote + ENN | Smote + Tomek | Smote | *Over* aleatório | NCL | *Under* aleatório | Tomek | OSS∇ | CNN∇ | Original∇ | CNN + Tomek∇ |
| 5 | *Over* aleatório | Original | Tomek | Smote | CNN | NCL | Smote + Tomek | Smote + ENN∇ | CNN + Tomek∇ | *Under* aleatório∇ | OSS∇ |
| 6 | Smote | Smote + Tomek | Smote + ENN | CNN + Tomek | OSS | *Over* aleatório | Tomek | CNN | NCL | Original | *Under* aleatório |
| 7 | *Over* aleatório | Smote | Smote + Tomek | OSS | CNN | Original | CNN + Tomek | Tomek | NCL∇ | Smote + ENN∇ | *Under* aleatório∇ |
| 8 | Smote + ENN | Smote + Tomek | Smote | *Over* aleatório | Tomek∇ | NCL∇ | Original∇ | CNN ∇ | CNN + Tomek∇ | *Under* aleatório∇ | OSS∇ |
| 9 | Smote + ENN | Smote + Tomek | Smote | *Over* aleatório | *Under* aleatório | CNN | Original | Tm k | CNN + Tomek | NCL | OSS |
| 10 | Smote + Tomek | Smote | Smote + ENN | *Over* aleatório | NCL | Tomek | *Under* aleatório | Original | OSS | CNN + Tomek∇ | CNN∇ |
| 11 | Smote + ENN | Smote | Smote + Tomek | *Over* aleatório | NCL | Tomek | Original∇ | OSS∇ | CNN + Tomek∇ | *Under* aleatório∇ | CNN∇ |
| 12 | *Over* aleatório | Smote + ENN | Smote + Tomek | CNN + Tomek | Smote | *Under* aleatório | CNN∇ | OSS∇ | Tomek∇ | Original∇ | NCL∇ |
| 13 | Smote + ENN | *Over* aleatório | NCL | Smote | Smote + Tomek | Original | Tomek | *Under* aleatório | CNN + Tomek∇ | OSS∇ | CNN∇ |
| 14 | Smote + Tomek | Smote + ENN | Smote | *Over* aleatório | OSS | Original | Tomek | CNN + Tm k | NCL | CNN | *Under* aleatório∇ |
| 15 | *Over* aleatório | Tomek | Original | NCL | CNN∇ | OSS∇ | Smote + Tomek∇ | Smote∇ | CNN + Tomek∇ | Smote + ENN∇ | *Under* aleatório∇ |

**Tabela 7.3:** *Ranking* de desempenho dos métodos de balanceamento artificial de conjuntos de dados para as árvores podadas. O conjunto original está destacado em cinza escuro. Os métodos de *over-sampling* em cinza claro. O símbolo ∇ indica desempenho estatisticamente inferior ao melhor resultado.

| # | 1º | 2º | 3º | 4º | 5º | 6º | 7º | 8º | 9º | 10º | 11º |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | *Over* aleatório | Smote | Smote + Tomek | Smote + ENN | Tomek | NCL | Original | CNN + Tomek | *Under* aleatório | CNN∇ | OSS∇ |
| 2 | Original | Tomek | *Over* aleatório | NCL | *Under* aleatório | Smote | Smote + Tomek | Smote + ENN | OSS | CNN | CNN + Tomek |
| 3 | Original | CNN + Tomek | *Over* aleatório | Smote | Tomek | CNN | OSS | Smote + ENN | *Under* aleatório | Smote + Tomek∇ | NCL∇ |
| 4 | Smote + Tomek | Smote + ENN | Smote | *Over* aleatório | Tomek | NCL | *Under* aleatório | Original | OSS∇ | CNN∇ | CNN + Tomek∇ |
| 5 | Original | Smote | Tomek | CNN | NCL | Smote + Tomek | *Over* aleatório | OSS∇ | Smote + ENN∇ | CNN + Tomek∇ | *Under* aleatório∇ |
| 6 | *Over* aleatório | Smote | Smote + Tomek | Smote + ENN | Original | CNN + Tomek | Tomek | NCL | OSS | CNN | *Under* aleatório |
| 7 | *Over* aleatório | Smote | Smote + Tomek | OSS | CNN | Original | CNN + Tomek | Tomek | Smote + ENN | NCL | *Under* aleatório∇ |
| 8 | Smote + ENN | Smote | Smote + Tomek | Tomek∇ | *Over* aleatório∇ | NCL∇ | Original∇ | CNN∇ | CNN + Tomek∇ | OSS∇ | *Under* aleatório∇ |
| 9 | Smote + ENN | Smote | Smote + Tomek | *Over* aleatório | Original | Tomek | *Under* aleatório | CNN | CNN + Tomek | NCL | OSS |
| 10 | Smote + Tomek | Smote | Smote + ENN | Tomek | *Over* aleatório | Original | NCL | *Under* aleatório | OSS | CNN + Tomek∇ | CNN∇ |
| 11 | Smote + ENN | Smote | Smote + Tomek | *Over* aleatório | Tomek | NCL | Original | CNN + Tomek∇ | *Under* aleatório∇ | OSS∇ | CNN∇ |
| 12 | Smote + Tomek | OSS | *Over* aleatório | CNN | Tomek | Smote + ENN | *Under* aleatório | Original | NCL | CNN + Tomek | Smote |
| 13 | Smote + ENN | *Over* aleatório | NCL | Smote | Smote + Tomek | Original | Tomek | *Under* aleatório | OSS∇ | CNN + Tomek∇ | CNN∇ |
| 14 | Smote | Smote + Tomek | Smote + ENN | *Over* aleatório | Tomek | OSS | NCL | Original | CNN | CNN + Tomek | *Under* aleatório∇ |
| 15 | *Over* aleatório | Original | NCL | Tomek | CNN | OSS∇ | Smote∇ | CNN + Tomek∇ | Smote + Tomek∇ | Smote + ENN∇ | *Under* aleatório∇ |

**Tabela 7.4:** *Ranking* de desempenho dos métodos de balanceamento artificial de conjuntos de dados para as árvores não podadas. O conjunto original está destacado em cinza escuro. Os métodos de *over-sampling* em cinza claro. O símbolo ∇ indica desempenho estatisticamente inferior ao melhor resultado.

mentada, tendo um comportamento similar. Em contrapartida, Japkowicz & Stephen (2002) comparam vários métodos diferentes de *over-sampling* e *under-sampling* em uma série de conjuntos de dados artificiais, e concluem que *over-sampling* é mais efetivo que *under-sampling* na redução do erro.

Em nossa opinião, os bons resultados de *over-sampling* não são completamente inesperados. Como visto anteriormente, existem evidências que o baixo desempenho está diretamente associado a uma baixa representatividade (em termos absolutos) dos exemplos das classes minoritárias, em conjunção com outros fatores complicantes, como o grau de sobreposição entre as classes. Os métodos de *over-sampling* são aqueles que atacam o problema da falta de exemplos da classe minoritária mais diretamente.

Como os resultados dos métodos de *over-sampling* apresentaram os melhores resultados, assim como as árvores não podadas, conduzimos uma análise para verificar o tamanho das árvores geradas, medido considerando o número de ramos e altura da árvore. Nas Tabelas 7.5 e 7.6 são mostrados, respectivamente, o número médio de ramos e a altura média da árvore para os métodos de *over-sampling* e as árvores originais não podadas. Os melhores resultados (menos ramos e menor altura da árvore) são mostrados em negrito, e as células que contêm os melhores resultados, não se levando em consideração o conjunto de dados original, são destacadas em cinza claro.

| Conj. dados | Original | *Over* Al. | SMOTE | SMOTE+Tomek | SMOTE+ENN |
|---|---|---|---|---|---|
| Pima | **29.90(6.06)** | 63.80(13.15) | 57.70(11.52) | 54.20(12.91) | 47.50(8.76) |
| German | 315.50(21.41) | 410.60(28.64) | 367.30(20.85) | 355.10(24.20) | **261.00(28.08)** |
| Post-operative | **20.40(3.86)** | 36.80(3.05) | 38.60(4.35) | 32.70(5.87) | 25.90(4.09) |
| Haberman | **7.80(3.79)** | 25.20(10.94) | 23.20(9.61) | 25.00(7.70) | 30.30(4.92) |
| Splice-ie | **203.50(7.78)** | 258.70(13.07) | 443.20(16.69) | 340.60(21.34) | 307.90(17.21) |
| Splice-ei | **167.80(9.40)** | 193.30(7.41) | 374.50(20.41) | 283.90(14.90) | 248.80(12.90) |
| Vehicle | **26.20(3.29)** | 28.90(2.60) | 34.90(3.38) | 34.90(3.38) | 29.20(2.82) |
| Letter-vowel | **534.50(11.92)** | 678.80(19.07) | 1084.50(19.61) | 1083.20(20.12) | 1022.00(26.34) |
| New-thyroid | 5.40(0.84) | **5.10(0.32)** | 6.90(1.29) | 6.90(1.29) | 6.90(0.99) |
| E-coli | **11.60(3.03)** | 17.70(2.91) | 16.70(3.20) | 16.50(3.84) | 12.70(3.23) |
| Satimage | **198.80(11.04)** | 252.70(9.33) | 404.60(12.97) | 404.60(12.97) | 339.40(13.80) |
| Flag | **28.60(6.52)** | 46.30(7.72) | 52.50(12.47) | 46.50(13.36) | 40.30(9.09) |
| Glass | **9.40(2.22)** | 13.00(1.33) | 17.70(1.77) | 17.70(1.77) | 15.50(1.58) |
| Letter-a | **59.10(3.45)** | 88.00(5.56) | 257.60(15.42) | 257.60(15.42) | 252.60(18.23) |
| Nursery | **229.40(4.65)** | 282.50(5.34) | 1238.30(28.91) | 1204.70(27.94) | 766.30(77.24) |

**Tabela 7.5:** Número de regras (ramos) para o conjunto de dados originais e após a aplicação de *over-sampling* para as árvores podadas. Valores em negrito indicam menor número de regras. Células destacadas em cinza claro indicam menor número de regras sem levar em consideração o conjunto original de exemplos.

Os resultados apresentados na Tabela 7.5 estão representados graficamente na Figura 7.4. Nessa figura pode-se observar que os métodos de *over-sampling* levam geralmente a árvores maiores, se comparados com as árvores induzidas a partir do conjunto de dados original. Esse resultado é esperado, uma vez que a aplicação desses métodos leva a um incremento do conjunto de treinamento, o que geralmente leva a árvores de decisão maiores. Comparando o número médio de ramos obtidos nas quais foram

| Conj. dados | Original | *Over* Al. | SMOTE | SMOTE+Tomek | SMOTE+ENN |
|---|---|---|---|---|---|
| Pima | **6.21(0.61)** | 7.92(0.64) | 7.74(0.44) | 7.59(0.54) | 7.27(0.67) |
| German | **6.10(0.17)** | 6.89(0.25) | 10.27(0.51) | 9.68(0.32) | 7.35(0.58) |
| Post-operative | **3.61(0.41)** | 4.86(0.26) | 5.36(0.37) | 4.75(0.52) | 4.46(0.50) |
| Haberman | **3.45(1.36)** | 5.71(1.43) | 5.61(1.27) | 5.81(1.02) | 6.45(0.60) |
| Splice-ie | 6.04(0.09) | 6.15(0.04) | 6.08(0.08) | 6.00(0.09) | **5.58(0.11)** |
| Splice-ei | 5.46(0.14) | 5.70(0.08) | 5.51(0.07) | 5.41(0.09) | **4.91(0.09)** |
| Vehicle | 7.21(0.70) | 7.03(0.44) | 7.09(0.50) | 7.09(0.50) | **6.63(0.38)** |
| Letter-vowel | 20.96(1.19) | 19.32(0.82) | 18.78(0.40) | 18.78(0.40) | **18.32(0.43)** |
| New-thyroid | **2.76(0.39)** | 2.85(0.17) | 3.12(0.26) | 3.12(0.26) | 3.08(0.20) |
| E-coli | 4.43(0.79) | 5.48(0.41) | 4.98(0.60) | 4.92(0.65) | **4.15(0.49)** |
| Satimage | **12.13(0.46)** | 15.93(0.42) | 13.89(0.64) | 13.89(0.64) | 12.54(0.36) |
| Flag | **3.92(0.70)** | 5.42(0.55) | 9.43(1.04) | 8.75(1.53) | 6.71(1.23) |
| Glass | **4.20(0.61)** | 5.80(0.51) | 5.92(0.50) | 5.92(0.50) | 5.51(0.32) |
| Letter-a | **7.30(0.22)** | 10.35(0.64) | 10.97(0.38) | 10.97(0.38) | 10.86(0.36) |
| Nursery | 6.51(0.01) | 6.84(0.03) | 6.87(0.03) | 6.84(0.03) | **6.41(0.12)** |

**Tabela 7.6:** Número de regras (ramos) para o conjunto de dados originais e após a aplicação de *over-sampling* para as árvores não podadas. Valores em negrito indicam menor número de regras. Células destacadas em cinza claro indicam menor número de regras sem levar em consideração o conjunto original de exemplos.

aplicados os métodos de *over-sampling*, SMOTE + ENN e *over-sampling* aleatório são os métodos que levaram a um menor aumento no tamanho da árvore. O fato de *over-sampling* aleatório estar entre os métodos de *over-sampling* que produziram as menores árvores é, no entanto, surpreendente, principalmente com relação aos métodos em que SMOTE é associado com ENN e Tomek. Essa associação foi proposta com o intuito de eliminar ruídos e exemplos junto à borda e, conseqüentemente, simplificar a árvore de decisão.

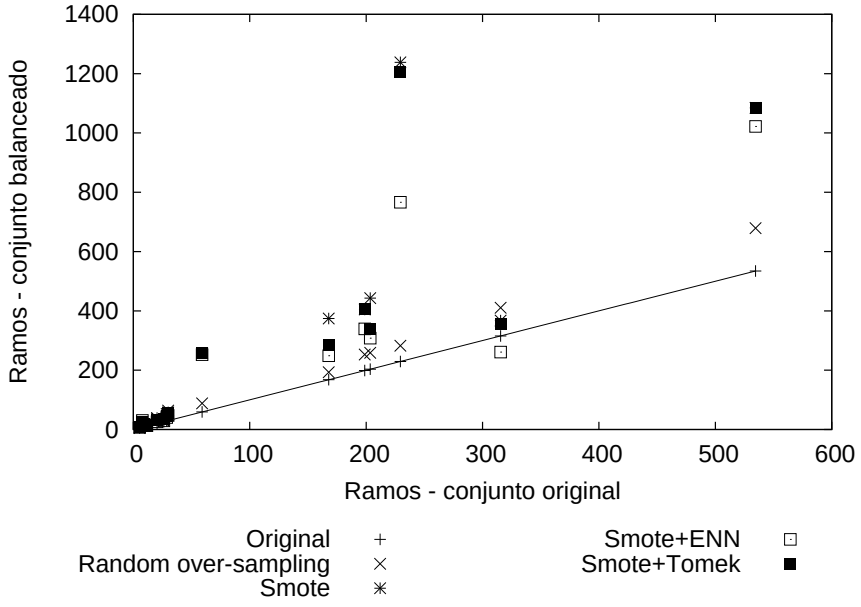



**Figura 7.4:** Número médio de condições por regra para os conjuntos originais e balanceados e árvores podadas.

O gráfico mostrado na Figura 7.5 representa os resultados da Tabela 7.6. Nesse gráfico é possível interpretar mais facilmente os resultados com respeito à altura da árvore. O método SMOTE + ENN teve resultados muito

bons, na maioria das vezes apresentado a árvore com menor altura. Esse método foi capaz inclusive de obter árvores com menor altura do que aquelas obtidas a partir do conjunto original de exemplos em 6 conjuntos de dados.

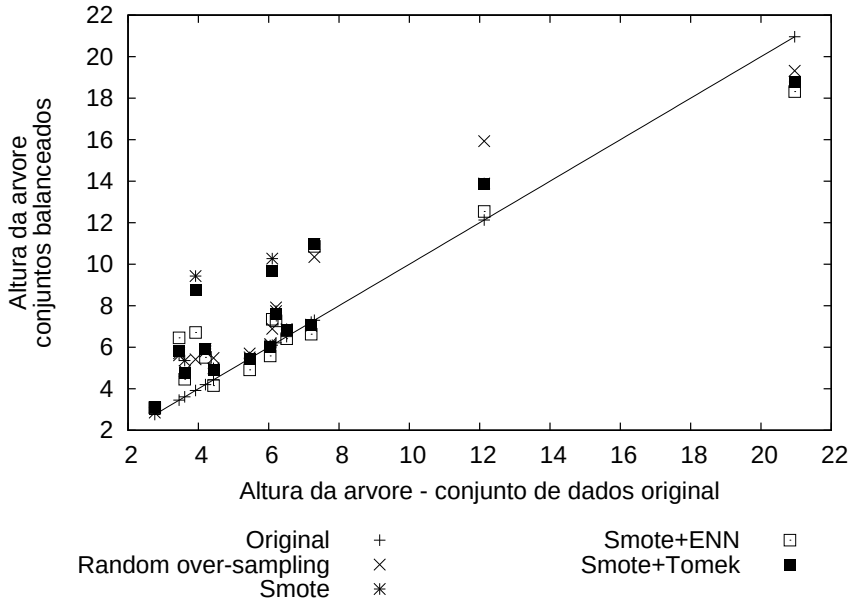



**Figura 7.5:** Número médio de condições por regra para os conjuntos originais e balanceados e árvores não podadas.

### 7.1.2 Avaliação experimental de proporções de exemplos entre as classes

Em uma outra série de experimentos desenvolvemos um estudo experimental para avaliar o comportamento que várias proporções diferentes de exemplos entre as classes têm sobre algoritmos de aprendizado. Utilizamos os dois métodos de amostragem aleatória: *over-sampling* e *under-sampling*. Nesse experimento, utilizamos 15 conjuntos de dados do repositório da UCI. Nesses conjuntos de dados, aplicamos os métodos de *over-sampling* e *under-sampling* aleatório até atingirmos treze diferentes distribuições de exemplos entre as classes pré-fixadas. Na Tabela 7.7 são apresentadas as principais características dos conjuntos de dados utilizados nesse experimento. Similarmente aos outros conjuntos utilizados neste trabalho, para conjuntos de dados com mais de duas classes, a classe com menos exemplos foi designada como positiva e o restante foi agrupado na classe negativa.

Nesse experimento removemos ou duplicamos exemplos do conjunto de treinamento aleatoriamente até atingirmos as seguintes distribuições de exemplos entre as classes: 5%, 7.5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 92.5% e 95%. Proporções maiores que 50% significam

| # | Conj. dados | $n_{ex}$ | $n_{atr}$ | Classe maj.% |
|---|---|---|---|---|
| 1 | sonar | 208 | 61 (61,0) | 53.5% |
| 2 | heart | 270 | 14 (14,0) | 55.6% |
| 3 | bupa | 345 | 7 (7,0) | 58.0% |
| 4 | ionosphere | 351 | 34 (34,0) | 64.1% |
| 5 | breast | 683 | 10 (10,0) | 65.0% |
| 6 | pima | 768 | 8 (8,0) | 65.2% |
| 7 | tic-tac-toe | 958 | 10 (0,10) | 65.3% |
| 8 | german | 1000 | 20 (7,13) | 70.0% |
| 9 | post-operative | 90 | 8 (1,7) | 73.3% |
| 10 | haberman | 306 | 3 (3,0) | 73.5% |
| 11 | vehicle | 846 | 18 (18,0) | 76.5% |
| 12 | new-thyroid | 215 | 5 (5,0) | 83.7% |
| 13 | ecoli | 336 | 7 (7,0) | 89.6% |
| 14 | flag | 194 | 28 (10,18) | 91.2% |
| 15 | glass | 214 | 9 (9,0) | 92.1% |

**Tabela 7.7:** Descrição dos conjuntos de dados utilizados para a avaliação da variação da proporção de exemplos entre as classes. $n_{ex}$ é o número de exemplos. $n_{atr}$ é o número de atributos. Valores entre pararêntesis indicam o número de atribudos discretos e contínuos, respectivamente. Classe maj.% é porcentagem de exemplos na classe majoritáira.

que os métodos de *over-sampling* e *under-sampling* tornaram a classe positiva (minoritária) mais freqüente do que a classe negativa. Além disso, para atingirmos distribuições mais desbalanceadas, duplicamos exemplos da classe negativa (majoritária) ou removemos exemplos da classe negativa, dependendo do método que estava sendo utilizado.

Neste experimento também foi utilizado o C4.5 com parâmetros padrão. Para obter estimativas de probabilidade mais confiáveis para cada folha, utilizamos o método de correção de probabilidades $m$ (Cussens, 1993), apresentada na Seção 3.6. O parâmetro $m$ foi ajustado de tal maneira que $b \times m = 10$, como sugerido por Elkan (2001b), na qual $b$ é a proporção de exemplos *a priori* no conjunto de treinamento da classe positiva.

Na Tabela 7.8 são apresentados o valor médio da AUC para as árvores de decisão induzidas pelo C4.5 nos conjuntos de dados depois da aplicação do *over-sampling* e *under-sampling* aleatório. A primeira coluna da Tabela 7.8 contêm o número do conjunto de dados. A coluna seguinte apresenta a proporção natural de exemplos positivos e negativos, seguida pelo valor médio da AUC para essa distribuição. As próximas treze colunas apresentam os valores médios da AUC para as proporções de exemplos pré-fixadas. Cada uma das linhas foi dividida em duas, sendo que cada primeira linha da divisão contêm os resultados da aplicação de *over-sampling* e a outra de *under-sampling*. Todos os valores dessa tabela foram obtidos utilizando-se

validação cruzada com 10 partições e os valores entre parêntesis referem-se aos respectivos desvios-padrão.

Para a realização de testes estatísticos foi utilizado o teste bicaudal de Dunnet (Hochberg & Tamhane, 1987; Hsu, 1996), que, como já mencionado, é indicado para múltiplas comparações com um controle. Para a realização desse teste, as árvores induzidas com a distribuição original de exemplos foram utilizadas como controle. Dessa maneira, para cada conjunto de dados, testamos a seguinte hipótese:

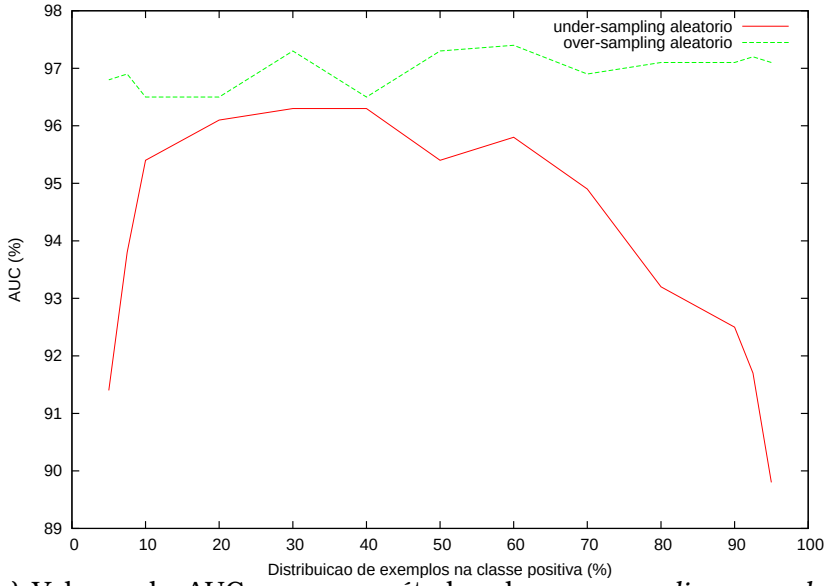
$$\begin{cases} H_0: & AUC_{\text{distribuição original}} = AUC_{\text{distribuições artificiais}} \\ H_1: & AUC_{\text{distribuição original}} \neq AUC_{\text{distribuições artificiais}} \end{cases}$$

O teste de hipótese foi realizado com um nível de confiança de 95%. Na Tabela 7.8, para as distribuições nas quais foi possível rejeitar $H_0$, *i.e.*, para distribuições em que há diferenças significativas entre os valores médios da AUC entre a árvore induzida com a distribuição original e as árvores induzidas com as distribuições artificiais, as células correspondentes estão destacadas em diferentes tons de cinza. As distribuições nas quais a AUC das árvores induzidas com as distribuições modificadas foram estatisticamente piores do que a árvore induzida com a distribuição original estão representadas por uma célula de cor cinza claro. Nas distribuições cujo desempenho das árvores induzidas com as distribuições artificiais são significativamente melhores do que as árvores induzidas com a distribuição original, as células foram coloridas de cinza escuro.

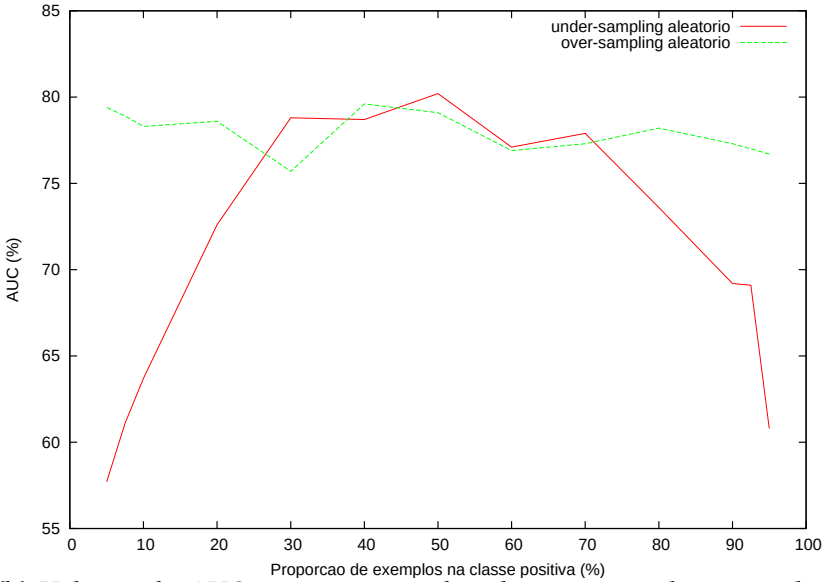
Observando essa tabela fica claro que o *over-sampling* e o *under-sampling* aleatório têm um comportamento muito diferente. Enquanto *over-sampling* não mostrou nenhuma diferença significativa com relação à distribuição original, *under-sampling* apresenta vários resultados em que a árvore induzida a partir do conjunto de dados em que esse método é aplicado é estatisticamente pior do que a distribuição original. A maioria desses resultados estão associados à distribuições muito desbalanceadas, com uma pequena tendência à esquerda, na qual a proporção de exemplos da classe positiva é maior. Outra importante diferença entre *over-sampling* e *under-sampling* aleatório é que os valores médios da AUC são bastante similares para várias distribuições e para vários conjuntos de dados. Entretanto, os melhores resultados do *under-sampling* aleatório concentram-se em distribuições mais balanceadas. Por exemplo, nas Figuras 7.6(a) e 7.6(b) são mostradas as AUC obtidas pelos métodos de *over-sampling* e *under-sampling* variando-se a proporção de exemplos para os conjuntos de dados Breast e Pima, respectivamente. Note que, em ambas as figuras, os valores da AUC para o

| # | Original | | Smp | Proportion of positive examples | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Prp | AUC | | 95% | 92.5% | 90% | 80% | 70% | 60% | 50% | 40% | 30% | 20% | 10% | 7.5% | 5% |
| 1 | 46.6 | 79.0(11.0) | *over* | 68.4(9.4) | 68.5(10.0) | 73.8(12.3) | 77.5(11.3) | 71.5(12.5) | 80.6(10.8) | 79.1(12.6) | 77.7(7.3) | 73.8(12.1) | 70.9(13.1) | 73.5(12.6) | 71.5(8.4) | 74.3(12.5) |
| | | | *under* | 53.3(10.6) | 55.0(12.3) | 56.0(8.4) | 75.7(17.3) | 77.5(15.6) | 72.7(17.9) | 78.4(10.6) | 78.8(14.9) | 70.6(12.4) | 65.9(9.9) | 66.8(12.2) | 65.5(11.7) | 62.3(10.0) |
| 2 | 44.4 | 85.5(10.4) | *over* | 82.2(14.3) | 84.5(14.3) | 87.3(10.1) | 88.1(10.3) | 87.2(9.4) | 84.1(10.0) | 84.2(9.0) | 85.6(11.0) | 87.6(9.1) | 84.7(10.3) | 86.1(12.4) | 87.6(9.4) | 87.3(8.6) |
| | | | *under* | 53.1(9.9) | 62.5(16.3) | 74.0(20.7) | 79.5(11.9) | 84.9(9.8) | 86.8(8.8) | 87.0(6.9) | 88.4(10.6) | 88.8(8.2) | 82.6(11.4) | 70.8(8.5) | 65.7(10.0) | 59.1(9.2) |
| 3 | 42.0 | 65.2(6.6) | *over* | 63.1(6.8) | 65.6(6.0) | 65.1(9.6) | 65.4(8.6) | 61.6(6.6) | 64.1(9.5) | 68.0(9.6) | 64.5(7.4) | 65.1(6.5) | 66.3(4.6) | 67.1(4.5) | 65.0(8.3) | 66.2(7.7) |
| | | | *under* | 50.0(0.0) | 49.6(1.5) | 52.7(7.1) | 57.1(7.2) | 61.3(10.8) | 61.8(9.1) | 64.0(6.6) | 62.9(6.2) | 61.1(7.4) | 55.4(8.6) | 57.3(9.2) | 53.2(7.9) | 50.8(2.5) |
| 4 | 35.9 | 91.4(5.0) | *over* | 89.4(6.6) | 89.7(5.8) | 91.8(4.6) | 92.2(4.9) | 91.9(4.3) | 92.7(4.7) | 92.8(6.4) | 92.7(3.7) | 91.3(5.1) | 92.9(4.1) | 90.2(5.3) | 90.9(5.8) | 89.4(6.6) |
| | | | *under* | 66.0(10.6) | 75.8(5.9) | 79.6(6.0) | 88.2(6.2) | 89.6(5.8) | 92.0(5.2) | 90.2(6.0) | 90.4(4.4) | 89.1(4.2) | 89.2(5.2) | 85.8(5.9) | 83.6(6.7) | 82.1(9.3) |
| 5 | 35.0 | 95.7(3.9) | *over* | 97.1(2.7) | 97.2(2.7) | 97.1(2.6) | 97.1(2.5) | 96.9(2.6) | 97.4(2.0) | 97.3(2.2) | 96.5(2.5) | 97.3(1.9) | 96.5(3.6) | 96.5(2.5) | 96.9(3.5) | 96.8(3.6) |
| | | | *under* | 89.8(4.3) | 91.7(5.0) | 92.5(3.8) | 93.2(2.9) | 94.9(2.7) | 95.8(2.1) | 95.4(2.7) | 96.3(3.4) | 96.3(2.2) | 96.1(2.9) | 95.4(3.5) | 93.8(5.3) | 91.4(6.3) |
| 6 | 34.8 | 79.0(5.1) | *over* | 76.7(7.0) | 77.0(6.7) | 77.3(7.5) | 78.2(7.3) | 77.3(8.2) | 76.9(4.5) | 79.1(5.3) | 79.6(6.8) | 75.7(6.9) | 78.6(5.3) | 78.3(7.2) | 78.9(6.9) | 79.4(5.2) |
| | | | *under* | 60.8(10.0) | 69.1(6.0) | 69.2(5.9) | 73.6(8.0) | 77.9(5.2) | 77.1(4.9) | 80.2(6.1) | 78.7(6.1) | 78.8(6.2) | 72.6(7.1) | 63.7(10.5) | 61.1(11.6) | 57.7(9.0) |
| 7 | 34.7 | 91.0(3.0) | *over* | 93.1(2.7) | 93.0(3.4) | 92.7(3.4) | 92.3(3.6) | 92.1(4.0) | 93.1(3.1) | 91.5(3.7) | 90.8(4.5) | 88.8(2.4) | 90.1(3.3) | 89.2(3.8) | 87.3(5.9) | 86.6(5.8) |
| | | | *under* | 50.0(0.0) | 51.7(5.2) | 53.4(6.0) | 76.0(6.3) | 78.3(5.8) | 83.8(5.6) | 87.8(5.4) | 87.8(4.4) | 84.9(4.5) | 75.7(7.9) | 69.6(9.4) | 59.5(10.9) | 52.4(5.0) |
| 8 | 30.0 | 72.4(3.9) | *over* | 69.9(6.1) | 70.9(5.8) | 69.1(4.8) | 72.2(5.0) | 70.9(4.0) | 71.8(6.4) | 72.5(5.9) | 71.3(5.9) | 72.6(4.0) | 72.1(4.1) | 73.2(4.3) | 71.9(4.0) | 72.4(4.5) |
| | | | *under* | 50.0(0.0) | 51.1(3.4) | 50.0(0.0) | 65.4(9.8) | 70.4(7.9) | 71.9(7.5) | 70.4(5.9) | 72.7(6.1) | 72.8(4.0) | 69.8(7.9) | 50.0(0.0) | 50.0(0.0) | 50.0(0.0) |
| 9 | 26.7 | 50.0(0.0) | *over* | 48.0(29.9) | 49.8(27.1) | 50.2(25.8) | 44.5(26.1) | 47.0(27.0) | 50.2(26.2) | 52.0(20.6) | 52.7(18.5) | 50.0(0.0) | 50.0(0.0) | 50.0(0.0) | 50.0(0.0) | 50.0(0.0) |
| | | | *under* | 50.0(0.0) | 50.0(0.0) | 50.0(0.0) | 50.0(0.0) | 50.0(0.0) | 46.9(9.9) | 48.2(9.8) | 42.7(9.3) | 50.0(0.0) | 50.0(0.0) | 50.0(0.0) | 50.0(0.0) | 50.0(0.0) |
| 10 | 26.5 | 54.1(7.9) | *over* | 60.5(7.2) | 62.6(9.2) | 62.9(7.8) | 59.7(9.9) | 62.5(10.0) | 63.8(10.3) | 64.1(12.2) | 61.3(10.6) | 64.3(14.7) | 51.2(5.3) | 54.9(13.2) | 52.9(7.4) | 52.6(8.7) |
| | | | *under* | 50.0(0.0) | 50.4(6.0) | 52.1(6.7) | 49.7(1.9) | 54.9(7.8) | 59.7(10.2) | 61.7(11.4) | 63.2(13.2) | 61.0(14.4) | 50.0(3.9) | 50.4(1.4) | 50.4(1.4) | 49.7(1.0) |
| 11 | 23.5 | 98.1(2.0) | *over* | 97.7(1.8) | 97.9(1.4) | 97.9(1.1) | 98.2(1.1) | 98.1(2.1) | 97.9(2.1) | 97.8(2.0) | 97.7(1.8) | 97.8(1.5) | 97.9(2.1) | 97.9(2.2) | 97.9(1.8) | 97.7(2.3) |
| | | | *under* | 75.6(7.0) | 84.7(9.0) | 87.5(7.8) | 94.6(3.8) | 95.3(2.8) | 96.5(2.9) | 97.3(2.5) | 97.3(2.4) | 97.5(1.8) | 97.8(1.9) | 97.5(2.3) | 96.5(2.6) | 94.0(3.3) |
| 12 | 16.3 | 89.0(18.8) | *over* | 91.2(18.9) | 91.2(18.9) | 91.2(18.9) | 91.2(18.9) | 91.2(18.9) | 90.6(18.7) | 90.6(18.7) | 90.6(18.7) | 90.3(18.5) | 84.3(21.1) | 87.9(18.2) | 85.9(19.0) | 92.6(10.7) |
| | | | *under* | 55.3(14.2) | 89.2(12.2) | 88.2(10.1) | 89.5(7.4) | 85.0(16.5) | 85.8(16.9) | 87.7(17.8) | 89.4(18.3) | 87.5(18.1) | 86.0(19.0) | 92.6(12.7) | 92.8(14.5) | 86.4(18.3) |
| 13 | 10.4 | 92.5(8.1) | *over* | 94.9(5.5) | 94.6(5.8) | 94.3(6.8) | 94.7(7.0) | 94.3(7.2) | 94.2(7.2) | 93.1(6.7) | 95.1(6.0) | 95.2(6.2) | 94.6(5.0) | 92.7(8.3) | 89.6(13.1) | 93.3(9.0) |
| | | | *under* | 69.8(13.9) | 70.8(13.1) | 72.9(10.1) | 84.2(7.0) | 85.8(5.6) | 89.3(6.7) | 93.1(4.2) | 94.0(3.9) | 93.0(2.9) | 85.4(14.3) | 94.3(4.4) | 83.5(15.5) | 80.3(20.0) |
| 14 | 8.8 | 50.0(0.0) | *over* | 66.2(22.8) | 67.8(24.2) | 67.0(23.3) | 69.3(25.4) | 66.5(26.5) | 66.7(27.1) | 63.4(17.5) | 63.1(21.5) | 55.8(22.8) | 60.0(22.9) | 49.2(1.9) | 50.0(0.0) | 50.0(0.0) |
| | | | *under* | 50.0(0.0) | 50.0(0.0) | 50.8(2.6) | 50.9(8.5) | 60.5(6.6) | 64.7(21.6) | 59.3(29.7) | 67.5(24.6) | 64.6(28.1) | 49.9(23.7) | 50.0(0.0) | 50.0(0.0) | 50.0(0.0) |
| 15 | 7.9 | 82.4(19.2) | *over* | 85.4(14.7) | 81.8(15.6) | 82.2(15.6) | 82.6(16.5) | 82.9(15.3) | 85.4(14.9) | 85.6(15.5) | 86.1(14.3) | 85.0(17.0) | 83.4(18.0) | 84.1(17.6) | 82.9(19.3) | 85.8(15.5) |
| | | | *under* | 50.0(0.0) | 50.0(0.0) | 56.5(6.5) | 57.6(6.3) | 63.9(19.3) | 65.3(23.0) | 67.1(16.8) | 74.4(22.1) | 69.3(22.0) | 79.0(19.8) | 83.4(18.0) | 62.9(21.3) | 51.7(5.3) |

**Tabela 7.8:** Resultados da AUC para conjuntos de dados após a aplicação de *under-sampling* e *over-sampling* até atingir diversas proporções de exemplos na classe positiva. Células destacadas em cinza claro indicam resultados piores do que os obtidos com o conjunto original. Céluas destacadas em cinza escuro indicam resultados melhores do que os obtidos com o conjunto original.




(a) Valores da AUC para os métodos de *over-sampling* e *under-sampling*. Conjunto de dados breast




(b) Valores da AUC para os métodos de *over-sampling* e *under-sampling*. Conjunto de dados pima

método de *over-sampling* estão bastante estáveis. No entanto, para o método de *under-sampling*, a linha pende para baixo nas pontas.

Comparando-se o desempenho dos métodos de *over-sampling* e *under-sampling* aleatório apresentados na Tabela 7.8, fica evidente que o *over-sampling* tem um desempenho geralmente melhor. Comparando-se os resultados em pares para cada conjunto de dados e para cada proporção de exemplos, *over-sampling* obteve resultados melhores em 164 dos 195 pares de resultados. Para 6 conjuntos de dados (bupa, ionosphere, breast, tic-tac-toe, vehicle e glass), *over-sampling* obteve valores de AUC superiores aos de *under-sampling* para todas as proporções. Para 11 dos 15 conjuntos de dados (sonar, bupa, ionosphere, breast, tic-tac-toe, german, post-operative,

vehicle, ecoli, flag e glass), o valor mais alto da AUC foi obtido por uma das distribuições na qual *over-sampling* foi aplicado.

As diferenças entre os dois métodos podem ser explicadas se levarmos em consideração que *over-sampling* duplica exemplos presentes no conjunto de dados, dessa maneira não removendo as informações associadas a esses exemplos no conjunto de dados. Em contrapartida, *under-sampling* aleatório remove exemplos para atingir a proporção de exemplos desejada, e as informações associadas aos exemplos removidos não é levada em consideração na indução do modelo.

## 7.2. Outros trabalhos relacionados ao aprendizado com proporções e exemplos entre as classes

Nesta seção são apresentados alguns outros trabalhos desenvolvidos relacionados com a proporção de exemplos entre as classes.

### 7.2.1 Desbalanceamento entre as classes e pequenos disjuntos

Em (Prati et al., 2004b) investigamos o relacionamento entre dois importantes tópicos alvos de recentes pesquisas em AM: aprendizado com classes desbalanceadas, já discutido nas seções anteriores deste capítulo, e pequenos disjuntos. Em linhas gerais, pequenos disjuntos podem ser entendidos como regras que cobrem corretamente um número reduzido de exemplos. Como visto anteriormente, o desbalanceamento entre as classes, associado a alguns outros fatores, pode levar a um baixo desempenho de algoritmos de aprendizado. Em contrapartida, pequenos disjuntos são freqüentemente reportados na literatura como tendo maiores taxas de erro do que grandes disjuntos.

Weiss (2003) sugere que existe uma relação entre pequenos disjuntos e o problema de desbalanceamento entre as classes. Além disso, Japkowicz (2003) concorda com essa hipótese e afirma que o problema de desbalanceamento entre as classes é potencializado quando o algoritmo se depara com pequenos disjuntos. Mesmo que esses trabalhos apontem uma conexão entre esses dois problemas, o relacionamento entre eles ainda não está bem estabelecido. A seguir reportamos alguns experimentos que realizamos tentando explorar esse relacionamento.

Uma conexão direta entre os temas pode ser traçada se observarmos que, uma vez que as classes minoritárias têm poucos exemplos associados a elas, regras que cobrem essas classes cobrirão um pequeno número de

exemplos. Além disso, essas regras provavelmente terão taxas de erro mais altas, uma vez que em problemas com classes desbalanceadas o número de exemplos da classe minoritária também é pequeno nos conjuntos de teste. No entanto, em presença de desbalanceamento de classes, como uma das estratégias mais utilizadas é maximizar o erro global, algoritmos de aprendizado podem "ignorar" classes com poucos exemplos (por exemplo, utilizando métodos de poda) e favorecer a indução de disjuntos maiores. Em outras palavras, problemas com classes desbalanceadas podem, dependendo de vários fatores, tentar favorecer a indução de pequenos quanto de grandes disjuntos.

Também é importante apontar as diferenças entre os dois problemas: classes com poucos exemplos existem na população em que os exemplos de treinamento são retirados, mas pequenos disjuntos podem ser uma conseqüência de algumas preferências do algoritmo. Na verdade, como dito anteriormente, classes desbalanceadas podem ou não formar pequenos disjuntos, mas pequenos disjuntos podem aparecer mesmo que a proporção de exemplos entre as classes seja igualmente balanceada (Weiss, 1995).

### Curva de concentração de erro

Para avaliar o grau em que erros estão concentrados nos pequenos disjuntos, Weiss (2003) propõe o uso da curva de concentração de erro. Para cada disjunto de tamanho $n$, para $n$ variando de 1 até o tamanho máximo dos disjuntos, constrói-se um gráfico com a porcentagem de erros no conjunto de teste *versus* a porcentagem de acertos. A linha $y = x$ corresponde a modelos nos quais os erros estão igualmente distribuídos pelos disjuntos. A concentração de erro (EC) é definida como a porcentagem da área total entre a linha $y = x$ e a curva da concentração de erro, sendo que valores abaixo dessa linha são negativos. A medida EC varia entre -100% até 100%; um valor negativo indica que os erros estão concentrados nos grandes disjuntos, enquanto que valores positivos indicam o contrário.

### Experimentos

Para avaliar o relacionamento entre as classes desbalanceadas e pequenos disjuntos, realizamos uma série de experimentos com vários conjuntos de dados da UCI (Newman et al., 1998). Assim como nas seções anteriores, na Tabela 7.9 são apresentadas as principais características dos conjuntos de dados utilizados nesse experimento.

Nessa série de experimentos, também utilizamos o algoritmo C4.5. Para cada conjunto de dados, calculamos a AUC e o EC para ambas as árvores podadas e não podadas, utilizando validação cruzada com 10 partições. Na

| Conj. dados | $n_{ex}$ | $n_{atr}$ | Classe maj.% |
|---|---|---|---|
| Sonar | 207 | 60 (60, 0) | 53,63% |
| Bupa | 345 | 6 (6, 0) | 57.97% |
| Pima | 768 | 8 (8, 0) | 65.23% |
| German | 1000 | 20 (7, 13) | 70.00% |
| Haberman | 306 | 3 (3, 0) | 73.53% |
| New-thyroid | 215 | 5 (5, 0) | 83.72% |
| E-coli | 336 | 7 (7, 0) | 89.58% |
| Satimage | 6435 | 36 (36, 0) | 90.27% |
| Flag | 194 | 28 (10, 18) | 91.24% |
| Glass | 214 | 9 (9, 0) | 92.06% |

**Tabela 7.9:** Descrição dos conjuntos de dados utilizados para averiguar a relação entre pequenos disjuntos e a proporção de exemplos entre as classes. $n_{ex}$ é o número de exemplos. $n_{atr}$ é o número de atributos. Valores entre pararêntesis indicam o número de atribudos discretos e contínuos, respectivamente. Classe maj.% é porcentagem de exemplos na classe majoritáira.

| Conjunto de Dados | Árvore podada | | Árvore não podada | |
|---|---|---|---|---|
| | AUC | EC | AUC | EC |
| Sonar | 86.71(6.71) | 61.51(19.03) | 86.71(6.71) | 61.51(19.03) |
| Bupa | 79.44(4.51) | 66.03(12.36) | 79.93(5.02) | 65.80(14.04) |
| Pima | 81.53(5.11) | 42.03(11.34) | 82.33(5.70) | 45.41(8.52) |
| German | 78.49(7.75) | 52.92(17.22) | 85.67(4.37) | 87.61(7.72) |
| Haberman | 58.25(12.26) | 29.33(22.51) | 67.91(13.76) | 36.25(20.06) |
| New-thyroid | 94.73(9.24) | 33.54(41.78) | 94.98(9.38) | 33.13(42.64) |
| E-coli | 87.64(15.75) | 55.13(36.68) | 92.50(7.71) | 71.97(26.93) |
| Satimage | 93.73(1.91) | 80.97(4.19) | 94.82(1.18) | 83.75(5.21) |
| Flag | 45.00(15.81) | 0.00(0.00) | 76.65(27.34) | 61.82(39.01) |
| Glass | 88.16(12.28) | 56.53(57.38) | 88.16(12.28) | 56.53(57.38) |

**Tabela 7.10:** Valores da AUC e da EC para as árvores podadas e não podadas no conjunto original de exemplos.

Tabela 7.10 são apresentados os resultados desse experimento. Os valores entre parêntesis indicam os respectivos desvios padrão. Para dois dos conjuntos de dados, Sonar e Glass, o C4.5 não podou a árvore.

Consideramos ambas as árvores podadas e não podadas pois estamos interessados em analisar se a poda é efetiva para tratar o problema de pequenos disjuntos na presença de desbalanceamento entre as classes. Poda é geralmente reportado na literatura como um método efetivo para a remoção de pequenos disjuntos indesejados. No entanto, como discutido nas seções anteriores, a poda não é benéfica na presença de classes desbalanceadas ou quando os custos de classificação são desconhecidos. Analisando os valores da AUC na Tabela 7.10 notamos que as árvores não podadas sistematicamente apresentam resultados melhores do que as árvores podadas.

Entretanto, também é possível observar que os valores da EC também aumentam para quase todas as árvores não podadas, *i.e.*, os erros de clas-

| Conjunto de Dados | Random | | SMOTE | |
|---|---|---|---|---|
| | AUC | EC | AUC | EC |
| Sonar | 86.52(4.69) | 47.29(27.24) | 86.74(8.91) | 52.07(24.63) |
| Bupa | 80.06(3.48) | 33.14(26.01) | 72.81(9.13) | 40.47(23.94) |
| Pima | 86.03(4.14) | 57.59(17.65) | 85.97(5.82) | 52.62(13.18) |
| German | 85.03(4.91) | 84.07(4.55) | 84.19(5.54) | 81.95(12.18) |
| Haberman | 73.58(14.22) | 54.66(22.37) | 75.45(11.02) | 43.15(25.55) |
| New-thyroid | 98.89(2.68) | 15.71(40.35) | 98.91(1.84) | 23.83(38.53) |
| E-coli | 93.55(6.89) | 81.93(13.09) | 95.49(4.30) | 91.48(16.12) |
| Satimage | 95.52(1.12) | 86.81(3.23) | 95.69(1.28) | 90.35(3.02) |
| Flag | 79.78(28.98) | 85.47(16.41) | 73.87(30.34) | 54.73(44.75) |
| Glass | 92.07(12.09) | 81.48(22.96) | 91.27(8.38) | 78.17(30.85) |

**Tabela 7.11:** Valores da AUC e da EC para as árvores podadas e não podadas após a aplicação de *over-sampling* aleatório e SMOTE.

sificação nesse caso tendem a se concentrar nos pequenos disjuntos. Esses resultados apontam um compromisso entre podar ou não a árvore: por um lado, podar a árvore leva a disjuntos maiores e a uma menor concentração de erro; por outro lado, não podar a árvore e reter os pequenos disjuntos leva a um ganho de desempenho em termos da AUC. Em outras palavras, a poda prioriza regras gerais que tendem a predizer a classe majoritária, enquanto que a não poda prioriza regras menores mas que cobrem melhor os exemplos da classe minoritária.

Dando continuidade ao experimento, analisamos o comportamento dos métodos de balanceamento artificial do conjunto de dados e pequenos disjuntos. Weiss (2003) também investiga esse problema aplicando o método de *under-sampling* aleatório para balancear artificialmente os conjuntos de treinamento. Seus resultados mostram que as árvores induzidas a partir de conjuntos balanceados daquela maneira tendem a produzir menores valores da EC, com um pequeno ganho em termos da AUC. Essa redução no valor da EC pode ser explicada pela redução no número de disjuntos induzidos, o que é uma característica dos métodos de *under-sampling*. Entretanto, assim como a poda, essa abordagem pode remover do modelo pequenos disjuntos que sejam interessantes. Além disso, como discutido nas seções anteriores, os métodos de *over-sampling* tendem a produzir melhores resultados do que os métodos de *under-sampling* em termos da AUC. Por esses motivos, em nossos experimentos, aplicamos métodos de *over-sampling* para testar o comportamento desses métodos com relação à EC. Na Tabela 7.11 são mostrados os valores da AUC e da EC para dois métodos de *over-sampling* propostos na literatura: aleatório e SMOTE.

Os resultados mostrados na Tabela 7.11 são reportados para árvores não podadas. Esses resultados apresentam as mesmas tendências daqueles apresentados na Tabela 7.10. Os valores da AUC são em geral maiores do que aqueles obtidos a partir da árvore podada construída a partir do

conjunto de dados original, mas com maiores EC. Entretanto, apesar da tendência no aumento da EC quando se aplica *over-sampling*, esses valores são menores do que o EC da árvore não podada induzida a partir do conjunto original de exemplos. Além disso, em três conjuntos de dados (Sonar, Bupa e New-thyroid), SMOTE produziu EC menores do que a árvore podada induzida no conjunto original.

Esses resultados podem ser explicados observando-se que, uma vez que o SMOTE utiliza um método de interpolação, SMOTE pode ajudar na definição das fronteiras de decisão para cada uma das classes. Entretanto, esse método de interpolação pode introduzir ruído no conjunto de dados, levando a um aumento no número de pequenos disjuntos indesejáveis. Em outras palavras, mesmo que o SMOTE ajude a superar o problema com o desbalanceamento entre as classes, ele pode aumentar o problema dos pequenos disjuntos. Nesse sentido, também investigamos como a associação dos métodos de limpeza de dados ao SMOTE, SMOTE + ENN e SMOTE + Tomek (ver Seção 6.1.2), por nós propostos em Batista et al. (2004), se comportam com relação à EC. Nossa suposição é que os métodos de limpeza de dados poderiam ajudar a remover pequenos disjuntos indesejáveis, além da melhora de desempenho em termos da AUC.

Na Tabela 7.12 são mostrados os resultados desses métodos nos mesmos conjuntos de dados. Comparando ambos os métodos, pode ser observado que SMOTE + Tomek produziu um AUC maior em cinco conjuntos de dados (Sonar, Pima, German, Haberman e E-coli) e o SMOTE + ENN é melhor em dois conjuntos (Bupa e Glass). Para os outros conjuntos, o desempenho dos dois métodos é comparável (com uma diferença menor que 1%). Além disso, pode ser observado que em 3 conjuntos de dados (New-thyroid, Satimage e Glass) SMOTE+Tomek obteve resultados idênticos ao SMOTE (Tabela 7.11). Nesses casos não foram encontradas ligações Tomek.

Para uma melhor comparação dos resultados, na Tabela 7.13 é mostrado um *ranking* dos valores da AUC e EC obtidos para todos os métodos de *over-sampling* e árvores não podadas. Nessa tabela, O indica o conjunto original— Tabela 7.10; R e S representam respectivamente *over-sampling* aleatório e SMOTE — Tabela 7.11; e S+E e S+T representam SMOTE + ENN SMOTE + Tomek; $\surd_1$ indica que o método teve o melhor resultado e $\surd_2$ o segundo melhor para o conjunto de dados correspondente. Resultados com diferenças menores que 1% aparecem em conjunto.

Note que o método conjugado SMOTE + Tomek teve resultados de AUC entre os melhores para 7 conjuntos de dados (sonar, pima, haberman, new-thyroid, e-coli, satimage e flag). Além disso, em outros dois conjuntos, o seu desempenho ficou entre os segundo melhores (german e glass). Esse método

| Conjunto de Dados | SMOTE + ENN | | SMOTE + Tomek | |
|---|---|---|---|---|
| | AUC | EC | AUC | EC |
| Sonar | 85.31(11.09) | 52.56(28.21) | 86.90(9.62) | 49.77(17.24) |
| Bupa | 78.84(5.37) | 41.72(14.68) | 75.35(10.65) | 38.39(18.71) |
| Pima | 83.64(5.35) | 54.07(19.65) | 85.56(6.02) | 47.54(21.06) |
| German | 82.76(5.93) | 82.21(10.52) | 84.40(6.39) | 88.53(6.54) |
| Haberman | 77.01(5.10) | 62.18(19.08) | 78.41(7.11) | 43.26(29.39) |
| New-thyroid | 99.22(1.72) | 27.39(44.34) | 98.91(1.84) | 23.83(38.53) |
| E-coli | 95.29(3.79) | 87.58(18.36) | 95.98(4.21) | 90.92(16.17) |
| Satimage | 96.06(1.20) | 88.56(3.31) | 95.69(1.28) | 90.35(3.02) |
| Flag | 78.56(28.79) | 78.78(20.59) | 82.06(29.52) | 70.55(38.54) |
| Glass | 93.40(7.61) | 80.14(30.72) | 91.27(8.38) | 78.17(30.85) |

**Tabela 7.12:** Valores da AUC e da EC para as árvores podadas e não podadas após a aplicação de SMOTE + ENN e SMOTE + Tomek.

| Datasets | AUC | | | | | EC | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | O | R | S | S+E | S+T | O | R | S | S+E | S+T |
| Sonar | $\surd_1$ | $\surd_1$ | $\surd_1$ | $\surd_2$ | $\surd_1$ | | $\surd_1$ | | | $\surd_2$ |
| Bupa | $\surd_1$ | $\surd_1$ | | $\surd_2$ | | | $\surd_1$ | | | $\surd_2$ |
| Pima | | $\surd_1$ | $\surd_1$ | $\surd_2$ | $\surd_1$ | $\surd_1$ | | | | $\surd_2$ |
| German | $\surd_1$ | $\surd_1$ | | $\surd_2$ | $\surd_2$ | | $\surd_2$ | $\surd_1$ | $\surd_1$ | |
| Haberman | | | | $\surd_2$ | $\surd_1$ | $\surd_1$ | | $\surd_2$ | | $\surd_2$ |
| New-thyroid | $\surd_2$ | $\surd_1$ | $\surd_1$ | $\surd_1$ | $\surd_1$ | | $\surd_1$ | $\surd_2$ | | $\surd_2$ |
| E-coli | | $\surd_2$ | $\surd_1$ | $\surd_1$ | $\surd_1$ | $\surd_1$ | $\surd_2$ | | | |
| Satimage | $\surd_2$ | $\surd_1$ | $\surd_1$ | $\surd_1$ | $\surd_1$ | $\surd_1$ | $\surd_2$ | | | |
| Flag | | $\surd_2$ | | | $\surd_1$ | $\surd_2$ | | $\surd_1$ | | |
| Glass | | $\surd_2$ | $\surd_2$ | $\surd_1$ | $\surd_2$ | $\surd_1$ | | $\surd_2$ | | $\surd_2$ |

**Tabela 7.13:** *Ranking* da AUC e da EC para as árvores não podadas.

não ficou nenhuma vez entre as menores EC, mas para 6 conjuntos (sonar, bupa, pima, haberman, newthyroid e glass) ele ficou entre os segundos melhores valores da EC, sendo que em quatro desses conjuntos SMOTE + Tomek teve a maior AUC (sonar, pima, haberman e new-thyroid). Outro resultado interessante foi o conseguido pelo *over-sampling* aleatório.

### 7.2.2 Analisando a sensitividade na proporção de rotulação de exemplos do algoritmo Co-training

Aprendizado semi-supervisionado surgiu recentemente como uma alternativa para problemas nos quais não se tem um número razoável de exemplos para se realizar aprendizado supervisionado. Nesse tipo de aprendizado, é fornecido aos algoritmos uma pequena quantidade de exemplos rotulados e vários exemplos não rotulados; o objetivo é utilizar os exemplos não rotulados para melhorar os modelos induzidos. Um dos principais algoritmos de aprendizado semi-supervisionado é o algoritmo Co-training, proposto por Blum & Mitchell (1998). O método utilizado pelo Co-training consiste da indução de dois modelos, cada um deles induzido utilizando uma visão diferente dos exemplos, os quais cooperam entre si. Exemplos são rotulados somente se esses modelos tiverem um alto grau de certeza a respeito da classificação desses exemplos. Esses novos exemplos rotulados são então adicionados ao conjunto original de exemplos rotulados de ambos os algoritmos e o processo é repetido até não ser possível rotular exemplos com um alto grau de certeza.

Um parâmetro muito importante do algoritmo Co-training refere-se à proporção de exemplos de cada classe que são rotulados em cada iteração do algoritmo. Uma suposição muito comum nos algoritmos de aprendizado é que o conjunto de treinamento possui a mesma proporção de exemplos encontrada no mundo real. Essa proporção é geralmente estimada a partir do conjunto de exemplos de treinamento. Entretanto, em aprendizado semi-supervisionado, o conjunto de exemplos rotulados é pequeno. Dessa maneira, estimar a proporção de exemplos entre as classes usando somente esse pequeno número de exemplos é muito arriscado e, dificilmente, será obtida uma boa aproximação. Em (Matsubara et al., 2006) foram feitos alguns experimentos variando a proporção de exemplos rotulados em cada classe a cada iteração do algoritmo Co-training.

Para essa avaliação foram utilizados três diferentes conjuntos de textos: um subconjunto de artigos de notícias do UseNet (Newman et al., 1998), resumos de artigos científicos, títulos e referências coletados da série *Lecture Notes in Artificial Intelligence* (Melo et al., 2003) e páginas web e links relacionados a cursos de ciência da computação (Blum & Mitchell, 1998). Para

o primeiro conjunto de textos foram extraídos um subconjunto de 100 textos dos grupos de notícias `sci.crypt`, `sci.electronics`, `sci.med`, `sci.¬space`, `talk.politics.guns`, `talk.politics.mideast`, `talk.politics¬.misc` e `talk.religion.misc`. Os textos dos 4 primeiros grupos de notícias foram agrupados na classe `sci` (400 - 50%) e os textos dos outros grupos foram agrupados na classe `talk` (400 - %50). O conjunto LNAI contêm textos de 396 artigos de *Case Based Reasoning* (277 - 70%) e *Inductive Logic Programming* (119 - 30%). As duas visões para esses conjuntos foram obtidas utilizando-se a ferramenta PRETEXT[3], sendo que a primeira visão consiste em palavra stemizadas simples e a segunda em palavras duplas (Matsubara et al., 2005). O conjunto de dados COURSE [4] consiste de 1038 páginas web coletadas de vários sítios de departamentos de ciência da computação. As duas visões que compõem esse conjunto consistem nas palavras que aparecem na página e nas palavras que aparecem nas ligações que apontam para aquela página. As páginas são classificadas como `course` (221 - 20%) e `non-course` (817 - 80%). As principais características desses conjuntos de textos, o erro médio obtido pelo algoritmo *Naïve Bayes* em cada classe, bem como o erro médio geral calculado utilizando-se validação cruzada com 10 partições, são mostrados na Tabela 7.14.

| Conj. de Dados | Número de documentos | Visão | #Stem | $n_a tr$ | Classe | Erro Classe | Erro Total |
|---|---|---|---|---|---|---|---|
| NEWS | 800 | 1-gram | 15711 | 8668 | `sci` (50%)<br>`talk` (50%) | 2.5 (1.7)<br>0.8 (1.2) | 1.6 (1.0) |
| | | 2-gram | 71039 | 4521 | `sci` (50%)<br>`talk` (50%) | 2.0 (2.0)<br>0.5 (1.1) | 1.3 (1.2) |
| LNAI | 396 | 1-gram | 5627 | 2914 | ILP (30%)<br>CBR (70%) | 1.7 (3.7)<br>1.4 (1.9) | 1.5 (1.8) |
| | | 2-gram | 21969 | 3245 | ILP (30%)<br>CBR (70%) | 1.8 (1.7)<br>1.5 (1.9) | 1.8 (1.7) |
| COURSE | 1038 | TEXT | 13198 | 6870 | `course` (20%)<br>`non-course` (80%) | 16.3 (5.4)<br>3.8 (2.0) | 6.5 (2.3) |
| | | LINKS | 1604 | 1067 | `course` (20%)<br>`non-course` (80%) | 9.6 (7.6)<br>16.0 (4.7) | 14.6 (3.5) |

**Tabela 7.14:** Descrição dos conjuntos de textos utilizados no experimento com Co-training.

Todos os experimentos foram conduzidos com o mesmo número de exemplos no conjunto inicial de exemplos rotulados (30 exemplos) igualmente distribuídos entre as classes (15 exemplos de cada classe). A cada iteração, até 10 exemplos poderiam ser utilizados para incrementar o conjunto de exemplos rotulados. Utilizamos o limiar de 60% de confiança para rotular um exemplo em ambas as visões. Para analisar o impacto da proporção em que exemplos são rotulados em cada iteração, exemplos foram rotulados

[3] `http://www.icmc.usp.br/~edsontm/pretext/pretext.html`

[4] `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-51/www/co-training/data/`

nas seguintes proporções de exemplos em classes: 2/8, 3/7, 5/5, 7/3 e 8/2

Na Tabela 7.15 são mostrados os resultados médios obtidos no experimento. Todos os resultados foram obtidos utilizando-se validação cruzada com 10 partições. Para cada conjunto de textos, nessa tabela, as quatro primeiras linhas mostram o número de exemplos por classe que foram incorretamente (I) e corretamente (C) rotulados; L é o tamanho do conjunto final de exemplos rotulados; U' é o número de exemplos não rotulados pelo algoritmo; Erro e AUC são, respectivamente, o erro médio e a AUC do modelo final; e Errados é o número total de exemplos rotulados incorretamente. O melhor resultado para cada um desses valores está destacado em negrito.

| | 2/8 | 3/7 | 5/5 | 7/3 | 8/2 |
|---|---|---|---|---|---|
| NEWS dataset | | | | | |
| sci(I) | 18.00 (26.45) | 10.60 (15.47) | 1.10 (1.85) | 0.40 (0.52) | 0.80 (0.42) |
| sci(C) | 344.50 (2.72) | 339.40 (2.50) | 325.70 (11.51) | 203.60 (0.52) | 139.50 (1.51) |
| talk(I) | 1.60 (1.17) | 2.20 (0.63) | 5.70 (10.03) | 42.50 (30.34) | 131.00 (18.89) |
| talk(C) | 139.40 (1.17) | 201.80 (0.63) | 324.30 (10.03) | 345.70 (1.89) | 347.80 (3.08) |
| L | 503.50 (26.53) | 554.00 (15.30) | 656.80 (9.77) | 592.20 (30.07) | 619.10 (17.00) |
| U' | 206.50 (26.53) | 156.00 (15.30) | 53.20 (9.77) | 117.80 (30.07) | 90.90 (17.00) |
| Erro | 3.00 (3.24) | 2.38 (3.70) | **1.88 (2.14)** | 6.25 (5.14) | 19.00 (3.53) |
| AUC | 0.98 (0.02) | 0.98 (0.03) | **0.99 (0.02)** | 0.97 (0.04) | 0.92 (0.05) |
| Errados | 19.80 (26.96) | 12.80 (15.80) | **6.80 (11.77)** | 43.70 (30.29) | 133.50 (19.31) |
| LNAI dataset | | | | | |
| ilp(I) | 0.00 (0.00) | 1.30 (1.25) | 5.40 (1.71) | 9.30 (3.23) | 12.30 (5.10) |
| ilp(C) | 69.00 (0.00) | 94.20 (2.20) | 101.00 (1.49) | 100.80 (1.14) | 101.70 (1.57) |
| cbr(I) | 0.70 (0.95) | 0.00 (0.00) | 0.00 (0.00) | 0.00 (0.00) | 0.00 (0.00) |
| cbr(C) | 230.30 (0.95) | 204.00 (0.00) | 150.00 (0.00) | 96.00 (0.00) | 69.00 (0.00) |
| L | 300.00 (0.00) | 299.50 (1.08) | 256.40 (2.41) | 206.10 (3.54) | 183.00 (5.10) |
| U' | 50.00 (0.00) | 50.50 (1.08) | 93.60 (2.41) | 143.90 (3.54) | 167.00 (5.10) |
| Erro | **1.26 (1.33)** | 2.02 (2.00) | 2.03 (1.07) | 3.28 (1.69) | 4.80 (3.03) |
| AUC | **1.00 (0.00)** | 1.00 (0.01) | 0.99 (0.01) | 0.99 (0.01) | 0.99 (0.01) |
| Errados | **0.70 (0.95)** | 1.30 (1.25) | 5.60 (1.90) | 9.30 (3.23) | 12.50 (5.04) |
| COURSE dataset | | | | | |
| course(I) | 34.40 (29.73) | 103.90 (66.05) | 252.30 (72.89) | 423.40 (27.35) | 434.80 (112.58) |
| course(C) | 146.00 (26.82) | 132.80 (27.26) | 155.50 (13.34) | 175.40 (6.00) | 179.30 (10.89) |
| ncourse(I) | 5.30 (3.13) | 7.20 (8.00) | 4.20 (4.59) | 1.50 (2.92) | 2.40 (3.34) |
| ncourse(C) | 505.20 (154.07) | 307.10 (227.37) | 146.80 (110.20) | 81.60 (31.65) | 81.30 (56.98) |
| L | 690.90 (150.92) | 551.00 (186.16) | 558.80 (49.82) | 681.90 (23.39) | 697.80 (66.62) |
| U' | 239.10 (150.92) | 379.00 (186.16) | 371.20 (49.82) | 248.10 (23.39) | 232.20 (66.62) |
| Erro | **14.11 (13.26)** | 32.65 (20.15) | 49.43 (15.95) | 61.91 (8.07) | 60.29 (17.28) |
| AUC | **0.92 (0.08)** | 0.82 (0.11) | 0.71 (0.09) | 0.68 (0.07) | 0.67 (0.07) |
| Errados | **40.20 (31.71)** | 112.80 (67.28) | 258.70 (72.08) | 429.80 (25.59) | 442.60 (111.98) |

**Tabela 7.15:** Resultado da aplicação do Co-training para os conjuntos de dados NEWS, LNAI e COURSE. O melhor resultado de Erro, AUC e Errados está em negrito

Os resultados da aplicação do Co-training variando-se a proporção de exemplos entre as classes rotulados a cada iteração mostra um padrão interessante. Para o conjunto de dados naturalmente balanceado (NEWS), aumentando a proporção de exemplos rotulados na classe `talk`, *i.e*, rotulando-se exemplos na classe `talk` nas proporções: 7/2 e 8/2 não diminui em muito o desempenho. No entanto, aumentando-se a proporção de exemplos rotulados para a classe `sci`, o erro (de 1.88 na proporção 5/5 para 19.00 na proporção 8/2) bem como o número de exemplos incorretamente rotulados

(6.8 para 133.50 nas proporções 5/5 e 8/2) sobe drasticamente. Para os conjuntos que são desbalanceados, o padrão é mais claro: tanto a taxa de erro e o número de exemplos incorretamente rotulados aumentam quando a proporção de exemplos rotulados tende para proporções muito diferentes da natural.

Um outro resultado interessante está relacionado com a AUC. Para os conjuntos de dados com AUC altas, próximas de 100% — NEWS e LNAI, a degradação no desempenho é menor do que no outro conjunto (COURSE). Esse fato pode ser explicado pelo fato que uma AUC próxima de 100% é um forte indicativo de uma boa separação entre as classes, sendo dessa maneira mais fácil para o algoritmo construir modelos mais precisos.

## 7.3. Considerações finais

Neste capítulo foram apresentados diversos experimentos relacionados à porporção de exemplos entre as classes. Primeiramente, foram reportados experimentos com conjuntos de dados naturais e uma série de métodos que visam balancear artificialmente os conjuntos de dados. Nossos experimentos mostram que métodos de *over-sampling* tendem a ter um melhor desempenho, medido em termos da AUC, do que os métodos de *under-sampling*. Também verificamos que o método de *over-sampling* aleatório, apesar de extremamente simples, é bem competitivo na maioria dos casos com outros métodos de *over-sampling* mais sofisticados. Realizamos também uma série de experimentos variando-se artificialmente a proporção de exemplos dos conjuntos de dados, utilizando *over-sampling* e *under-sampling* aleatório. Mais uma vez o método de *over-sampling* teve melhor desempenho. O melhor desempenho dos métodos de *over-sampling* pode ser explicado principalmente devido ao fato que, por descartarem exemplos, os métodos de *under-sampling* tendem a descartar informações potencialmente contidas nesses exemplos. Também verificamos experimentalmente que, em geral, a poda da árvore de decisão leva a resultados com uma menor AUC, principalmente em conjuntos de dados desbalanceados.

Também conduzimos uma série de experimentos para explorar o relacionamento entre pequenos disjuntos e classes desbalanceadas. Em linhas gerais, pode-se concluir que os métodos de *over-sampling*, que mais diretamente tratam o problema de classes desbalanceadas, levam a um aumento no número e na concentração de erro dos pequenos disjuntos. Além disso, a poda, que é normalmente recomendada para diminuir a incidência de pequenos disjuntos, leva a uma diminuição da AUC para problemas com classes desbalanceadas. Um dos métodos por nós proposto, SMOTE + To-

mek, teve uma pequena melhora no desempenho conjunto tanto na AUC quanto na concentração de erro.

Com um outro experimento mostramos empiricamente que a proporção de exemplos de cada classe rotulados a cada iteração do algoritmo semi-supervisionado Co-training é um parâmetro muito importante desse algoritmo, fato normalmente negligenciado na literatura. A maioria dos experimentos com Co-training reportados na literatura refere-se a simulações nas quais a classe associada a cada exemplo é conhecida. Nesses experimentos, a proporção é normalmente ajustada à proporção de exemplos do conjunto de dados. Entretanto, essa informação é, a princípio, desconhecida.

Capítulo 8

# BordaRank: construindo *ensembles* de *rankings*

Neste capítulo é descrita uma nova abordagem para a criação de *ensembles* de *rankings*. Este capítulo está organizado da seguinte maneira: na Seção 8.1 são apresentadas algumas considerações iniciais ao tema. Na Seção 8.2 é apresentado o problema de *ranking* tratado neste trabalho. Na Seção 8.3 são apresentados alguns métodos de votação comumente utilizados. A nossa abordagem é inspirada em um desses métodos. Na Seção 8.4 são brevemente apresentados alguns trabalhos correlacionados. Nossa abordagem é descrita na Seção 8.5. Experimentos para a validação da abordagem proposta são descritos na Seção 8.6. Finalmente, na Seção 8.7 são apresentadas algumas considerações finais do capítulo.

## 8.1. Considerações iniciais

Vários métodos foram propostos na literatura para agregar a classificação de vários modelos de classificação em uma única classe. Esses métodos são geralmente conhecidos como *ensembles*, e cada modelo que compõe o *ensemble* é conhecido como modelo base. Em certas condições, *ensembles* proporcionam um ganho de desempenho com relação aos modelos base considerando a taxa de acerto de classificação (Dietterich, 2000). Alguns

exemplos de métodos de *ensemble* incluem bagging (Breiman, 1996) e boosting (Freund & Schapire, 1997), entre outros.

Bagging está baseado na geração de várias versões de um modelo, e na utilização dessas versões para se criar um modelo agregado. Geralmente utiliza-se um método de votação para combinar as predições. As versões do modelo são geralmente obtidas por métodos de amostragem com reposição do conjunto de exemplos de treinamento. Boosting, em contrapartida, é um algoritmo iterativo no qual a construção da combinação de modelos ocorre em estágios. A cada estágio, um modelo é induzido a partir dos dados. Esse modelo é então adicionado ao modelo combinado, com um peso proporcional ao quão preciso é o modelo. Também são atribuídos pesos aos exemplos, de maneira que os exemplos nos quais o modelo corrente erra a classificação recebem um peso maior — são "ampliados" (*boosted*) — de tal maneira que o próximo modelo a ser induzido tente classificar corretamente esses exemplos.

Mesmo que *ensembles* sejam largamente utilizados para melhorar modelos de classificação, pouco trabalho tem sido feito em métodos que constróem *ensembles* de *rankings*. Neste capítulo descrevemos um novo método que propomos para combinar vários *rankings* em um *ensemble*. O método é chamado BORDARANK e é baseado no método de votação por preferências múltiplas denominado borda count (Borda, 1781). Resultados experimentais, medidos em termos da AUC, mostram que BORDARANK tem um ganho de desempenho com relação aos *rankings* individuais que compõem o modelo. Além disso, BORDARANK é competitivo com métodos mais sofisticados que combinam predições contínuas, ao invés de apenas o *ranking* dos exemplos. Esse fato é uma das vantagens de BORDARANK, pois ele pode ser aplicado em problemas nos quais somente uma ordenação dos casos esteja disponível.

## 8.2. Ordenação

Neste trabalho, um *ranking* é definido como: dado um conjunto de casos e uma propriedade de interesse, geralmente relacionada a um atributo classe em problemas de classificação, um *ranking* é uma completa ordenação desses casos do mais provável ao menos provável de pertencer à classe de interesse. Em outras palavras, estamos interessados em ordenação binária, na qual gostaríamos de colocar o maior número possível de exemplos da classe de interesse no topo dos elementos ordenados. O problema de *ranking* é muito comum em aplicações reais, nas quais é necessário ordenar casos ao invés de apenas classificá-los. Um exemplo é um sistema de

recomendação, no qual o objetivo é obter uma lista ordenada de mercadorias — digamos livros ou filmes — que um consumidor gostaria de comprar, basedo em suas preferências. Outro exemplo é campanha de *marketing* direto ao consumidor por meio de catálogo de compras, na qual o vendedor gostaria de ter um lista ordenada de consumidores em potencial. Essa lista poderia ser utilizada para enviar correspondência a somente os consumidores que aparecem no topo da lista, maximizando o lucro esperado. Recuperação de informação também utiliza métodos de *ranking*, na qual o objetivo é ordenar um conjunto de documentos de acordo com algum critério de busca. O objetivo é colocar o maior número possível de documentos que casam com o critério de busca no topo da lista de documentos.

Ordenação é um meio termo entre classificação e regressão. Uma maneira direta de criar um *ranking* é utilizar algum algoritmo de aprendizado que prediga um valor contínuo, que de alguma maneira estime a confiança ou as chances de um certo caso pertencer à classe de interesse. O valor predito por esse algoritmo para cada um dos exemplos pode ser utilizado para ordenar os casos (Caruana et al., 1995; Cohen et al., 1999). Como o valor contínuo predito por esses algoritmos pode ser escalado para representar probabilidades, esses modelos são freqüentemente chamados de modelos de classificação probabilístico, mesmo que, estritamente falando, eles não predigam nem uma classe nem probabilidades. Para a ordenação, no entanto, os valores numéricos não são de interesse, uma vez que somente a ordem que eles definem é levada em consideração. Uma outra maneira de resolver o problema de ordenação é utilizar pares de exemplos de treinamento (Fürnkranz & Hüllermeier, 2003). No entanto, essa abordagem aumenta a complexidade dos dados de $O(n)$ para $O(n^2)$, na qual $n$ é o número de exemplos a serem ordenados. Outra abordagem é utilizar regressão ordinal (Herbrich et al., 1999), cujo objetivo é mapear os casos para um conjunto ordenado de posições numéricas.

Como descrito no Capítulo 4, uma propriedade interessante de *rankings* construídos dessa maneira é que eles podem ser facilmente transformados em modelos de classificação utilizando-se uma regra de ponto de corte, *i.e.*, ou predizendo os $x$ por cento casos do topo como da classe positiva[1], como no exemplo de correspondência direta ao consumidor, ou selecionando-se apropriadamente um limiar de classificação utilizando-se análise ROC. Além disso, também como descrito no Capítulo 4, a área abaixo da curva ROC, AUC, é numericamente equivalente ao teste de Wilcoxon e pode ser interpretada como a probabilidade que, dados dois casos, um de cada classe,

[1]Nesse capítulo, classe positiva designa os casos que têm a propriedade de interesse. O restante dos casos pertence à classe negativa.

selecionados aleatoriamente, o exemplo da classe positiva apareça primeiro no *ranking* do que o exemplo da classe negativa.

## 8.3. Métodos de votação

O conceito de votação é bem conhecido em ciências sociais, tais como política e economia. Métodos de votação incluem tanto escolha simples quanto escolha múltipla. Um dos métodos mais utilizados é a *votação simples com pluralidade*, no qual cada eleitor pode votar em um dos candidatos e o vencedor é aquele com mais votos. Variações incluem *votação por maioria*, na qual a eleição somente tem um vencedor no caso de um dos candidatos receber mais da metade dos votos e *votação em turnos*, na qual os dois candidatos com mais votos no primeiro turno avançam para o segundo. Um método de votação um pouco diferente é a *votação ponderada*, na qual eleitores expressam o seu grau de preferência por um candidato atribuindo um peso ao seu voto. Quanto mais alto o grau de confiança, mais o candidato é preferido pelo eleitor.

Métodos que combinam modelos de classificação consideram cada um dos modelos como um eleitor e as possíveis classes como candidatos. Na votação por escolha simples, *ensembles* de modelos de classificação usam amplamente votação por pluralidade ou votação ponderada. O vencedor da eleição é a classificação final do caso dada pelo *ensemble*. Na votação por pluralidade, cada modelo-base simplesmente vota em uma classe e na votação ponderada cada voto é ponderado com um peso que geralmente representa probabilidades ou distâncias.

Para votações de escolha múltipla, ao invés de cada eleitor selecionar apenas um candidato, eles são permitidos a expressar suas preferências para todos os candidatos. Isso pode ser feito de duas maneiras: ou expressando o grau de confiança (com um peso) para cada candidato ou simplesmente ordenando os candidatos, do mais ao menos preferido. Esses dois métodos, *múltiplas preferências por pesos* ou *ordenação*, necessitam de diferentes maneiras de selecionar um vencedor. Para o caso em que se utilizam pesos, calcula-se geralmente a soma ou produto dos pesos, e o vencedor é o candidato com o maior valor dessa soma ou produto. Quando se ordena os candidatos, um dos métodos mais utilizados para se eleger o vencedor é o *borda count* (Borda, 1781), no qual a posição média de todos os candidatos é calculada. Os candidatos são então reordenados pela suas posições médias e o candidato mais bem posicionado ganha a eleição.

Na combinação de modelos de classificação, para um dado exemplo, cada modelo ou ordena as possíveis classes (no caso de votação por ordenação)

ou expressa o seu grau de confiança para todas as classes (em votação por múltiplas preferências por pesos). A posição média ou os pesos combinados, respectivamente, são então utilizados para classificar o exemplo.

Métodos de múltiplas preferências por ordenação, como o borda count, utilizam mais informação que métodos de preferência simples. Entretanto, como eles consideram a confiança entre os candidatos em proporções fixas (as posições), eles não levam em consideração o grau de preferência entre os candidatos como os métodos de combinação de múltiplas preferências com pesos fazem. Dessa maneira, quando comparados com os métodos de múltiplas preferências com pesos, isso significa que existe uma perda de informação. Entretanto, métodos de ordenação são preferidos nos casos em que diferentes escalas são utilizadas para graduar candidatos, uma vez que eles evitam os problemas de incompatibilidade e escala entre os eleitores.

## 8.4. Trabalhos relacionados

Cohen et al. (1999) propõem um método para a combinação de *rankings* que minimiza diretamente o número de ordenações incorretas. O problema é transformado em um problema combinatorial de otimização de combinação de grafos de preferências, construídos a partir dos *rankings*. RankBoost (Freund et al., 2003) é uma adaptação do algoritmo clássico de boosting AdaBoost para problemas de *ranking*. Ele também usa grafos de preferências, mas tenta evitar a intratabilidade da otimização combinatorial do trabalho de Cohen et al. (1999). Recentemente Rudin et al. (2005) mostraram que para o problema de *ranking* de ordenação binária bi-partida, como a discutida nesta trabalho, AdaBoost e RankBoost são muito semelhantes. Em outras palavras, eles mostraram que os desempenhos de AdaBoost e RankBoost tendem a ser parecidos neste tipo de problema. Além disso, eles mostraram que se um modelo de classificação que sempre prediz uma das classes for adicionado aos modelos que formam a base do AdaBoost, AdaBoost e RankBoost têm um comportamento idêntico. Huang & Ling (2005) propuseram um método para reconstruir *ensembles* selecionando um subconjunto dos modelos que compõem o *ensemble*. Cada modelo é ponderado utilizando-se uma variação da AUC como peso, e aqueles com baixos valores de AUC são removidos do *ensemble*.

Vários trabalhos exploram diferentes métodos de votação para combinar modelos de classificação na construção de *ensembles*. van Erp et al. (2002) comparam 10 métodos de votação em 4 conjuntos de dados, entre eles votação por pluralidade, maioria, soma e produto de pesos e borda count para a escolha da melhor classificação de um exemplo em um *ensemble* do tipo

bagging (Breiman, 1996); soma e produto de pesos tiveram bons resultados para *ensembles* pequenos e borda count foi melhor em *ensembles* de tamanho maior. (Leung & Jr., 2003) conduziram um experimento mais extenso em 39 conjuntos de dados da UCI e com 5 métodos de votação diferentes, incluindo votação por pluralidade e borda count. Suas conclusões afirmam que diferentes métodos de votação podem melhorar o desempenho de *ensembles*, mas seus resultados não apontaram um único método de votação como melhor. Apesar disso, borda count teve bons resultados em domínios com ruído.

Mesmo que utilizados previamente em problemas de combinação de modelos de classificação, não é de nosso conhecimento a utilização de métodos de votação para a combinação de *rankings*. Na próxima seção apresentamos o algoritmo BORDARANK, um método para a construção de *ensembles* a partir de *rankings* inspirado no método de votação borda count. Para adaptar o borda count ao problema de *rankings*, ao invés de considerarmos cada uma das classes como candidatos, como em (van Erp et al., 2002; Leung & Jr., 2003), nosso método considera cada caso como um candidato. Após, aplicamos um método semelhante ao borda count para combinar os *rankings* individuais.

## 8.5. O algoritmo BORDARANK

Sempre que o borda count é utilizado para construir *ensembles* de modelos de classificação, cada uma das classes é um candidato; eleitores são os modelos base e devem prover uma ordenação de preferências das classes para cada um dos casos; e borda count é utilizado para escolher a classificação final para um caso em particular. Nossa abordagem, denominada BORDARANK, utiliza um método similar ao borda count para combinar vários *rankings* em um *ensemble* de *rankings*. Ao invés de considerar cada classe como candidato, nossa abordagem leva em consideração todos os casos a serem ordenados como possíveis candidatos. Cada eleitor é um *ranking* base que deve fornecer uma ordenação a respeito das chances de cada caso pertencer à classe de interesse, para todos os casos. Em caso de empates, uma posição média é atribuída para todos os casos empatados. Após isso, BORDARANK aplica uma abordagem similar ao borda count para combinar os *rankings* de todos os eleitores.

BORDARANK trabalha da seguinte maneira: ao invés de um conjunto de modelos base de classificação, temos um conjunto base de *rankings*; assuma que queiramos construir um *ensemble* de $m$ *ranking* base tendo $n$ casos ordenados em cada *ranking* base. O resultado pode ser representado

em uma tabela de $n \times m$ posições. Nessa tabela, a $j$-ésima coluna dá as posições do $j$-ésimo *ranking* base. Dessa maneira, a célula $(i, j)$ dá a posição do caso $i$ ordenado pelo *ranking* $j$. Para cada caso $i$, é calculada a média das suas posições individuais $i\_$. Finalmente, os casos são reordenados de acordo com as suas posições médias.

A aplicação de nossa abordagem é explicada no seguinte exemplo: suponha que queiramos combinar 4 *rankings* com 10 casos a serem combinados, como mostrado na Tabela 8.1. Nessa tabela, a primeira coluna (# caso) é o número do caso (que não reflete a ordenação verdadeira dos exemplos). A segunda coluna mostra a classe verdadeira de cada exemplo (se o caso pertence à classe positiva, ele é rotulado como +; caso contrário, como -). As colunas três a seis mostram, respectivamente, as posições dadas por quatro diferentes *rankings*. A coluna sete mostra a soma das posições e a coluna oito mostra a média das posições. Finalmente, a última coluna mostra a nova ordenação dos casos.

| # caso | classe | *ranking* | | | | Soma | Posição | Novo |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | posições | média | *ranking* |
| $c_1$ | + | 3 | 2 | 2 | 2 | 9 | 2.25 | 2 |
| $c_2$ | + | 2 | 1 | 3 | 1 | 7 | 1.75 | 1 |
| $c_3$ | + | 6 | 4 | 6 | 4 | 20 | 5 | 5 |
| $c_4$ | + | 1 | 3 | 1 | 6 | 11 | 2.75 | 3 |
| $c_5$ | + | 4 | 6 | 5 | 3 | 18 | 4.5 | 4 |
| $c_6$ | - | 5 | 5 | 4 | 7 | 21 | 5.25 | 6 |
| $c_7$ | - | 9 | 7 | 7 | 5 | 28 | 7 | 7 |
| $c_8$ | - | 10 | 8 | 8 | 9 | 35 | 8.75 | 9 |
| $c_9$ | - | 7 | 9 | 9 | 8 | 33 | 8.25 | 8 |
| $c_{10}$ | - | 8 | 10 | 10 | 10 | 38 | 9.5 | 10 |

**Tabela 8.1:** Calculando o *ranking* médio

Como pode ser visto na Tabela 8.1, cada *ranking* base deixa ao menos um exemplo negativo nas 5 primeiras posições do *ranking*. Por exemplo, o caso $c_6$, que é negativo, está posicionado na 5ª, 5ª e 4ª posições, respectivamente, pelos *rankings* base 1, 2 e 3. Já o caso $c_7$ aparece na 5ª posição no *ranking* base 4. Além disso, cada um dos *rankings* base posiciona um exemplo positivo nas 5 últimas posições do *ranking*.

Os *rankings* base 1, 2 e 4 têm uma AUC de 96%, e o *ranking* base 3 tem uma AUC de 92%. Além disso, modelos de classificação que predigam os 5 primeiros exemplos como positivos têm uma taxa de acerto de 80%. Entretanto, quando se computa a posição média, as ordenações erradas são corrigidas e o *ranking* combinado pelo BORDARANK tem uma AUC de 100%, também como o modelo de classificação que prediz os 5 primeiros exemplos como positivos tem uma taxa de acerto de 100%.

A suposição chave do método proposto é que, dado um número suficiente de *rankings* independentes, os possíveis posicionamentos errados dados pe-

los *rankings* individuais são corrigidos quando calculamos a posição média para cada caso. Em outras palavras, considerando a posição de um caso em particular como uma variável aleatória, quando mais *rankings* são adicionados ao *ensemble* mais a variância do *ranking* médio tende a zero, e a média das posições tende aos valores esperados associados de suas variáveis aleatórias.

## 8.6. Experimentos

Nesta seção são mostrados resultados experimentais utilizando 18 conjuntos de dados da UCI (Newman et al., 1998). Na Tabela 8.2 são apresentadas as principais características dos conjuntos de dados utilizados neste estudo.

| # | Conj. de dados | $n_{atr}$ | $n_{ex}$ | Classe maj. % |
|---|---|---|---|---|
| 1 | Breast | 9(9,0) | 683 | 65,00 |
| 2 | Bupa | 6(6,0) | 345 | 57,98 |
| 3 | Cleve | 13(6,7) | 296 | 54.05 |
| 4 | E.Coli | 7(7,0) | 336 | 89,58 |
| 5 | Flag | 28(10,18) | 194 | 91,24 |
| 6 | Flare | 10(2,8) | 1066 | 82.93 |
| 7 | German | 20(7,13) | 1000 | 70,00 |
| 8 | Glass | 9(9,0) | 214 | 92,07 |
| 9 | Haberman | 3(3,0) | 306 | 73.53 |
| 10 | Heart | 13(13,0) | 270 | 55.55 |
| 11 | Ionosphere | 33(33,0) | 351 | 64.10 |
| 12 | Kr-vs-Kp | 36(0,36) | 3196 | 52.22 |
| 13 | New-thyroid | 5(5,0) | 215 | 83.72 |
| 14 | Pima | 8(8,0) | 768 | 65.10 |
| 15 | Sonar | 60(60,0) | 208 | 53.37 |
| 16 | Tic-tac-toe | 9(0,9) | 958 | 65.34 |
| 17 | Tokyo | 44(44,0) | 959 | 63.92 |
| 18 | Vehicle | 18(18,0) | 846 | 76.48 |

**Tabela 8.2:** Descrição dos conjuntos de dados utilizados no experimento com BORDARANK. $n_{ex}$ é o número de exemplos. $n_{atr}$ é o número de atributos. Valores entre pararêntesis indicam o número de atribudos discretos e contínuos, respectivamente. Classe maj.% é porcentagem de exemplos na classe majoritáira.

Para validar nossa proposta, dividimos nossos experimentos em três estágios. Primeiramente, avaliamos se BORDARANK poderia melhorar o desempenho com relação a cada um dos *rankings* base. Para tanto, utilizamos vários modelos de classificação que predizem um valor contínuo (ou adaptamos algoritmos quando eles não predizem um valor continuo) e utilizamos esses valores para ordenar os casos. Após essa etapa, aplicamos BORDARANK para combinar esses *rankings* e realizamos testes estatísticos. No segundo estágio de nossos experimentos, comparamos BORDARANK com

vários métodos de combinar os valores contínuos preditos pelos algoritmos base para testar se BORDARANK é competitivo com essas abordagens. Finalmente, calculamos a precisão e a sensitividade dos modelos de classificação derivados utilizando os 5%, 10%, 20% e 30% dos casos que aparecem no topo dos *rankings*.

Para o primeiro estágio dos experimentos, para termos uma maior variabilidade nos resultados, selecionamos algoritmos com diferentes preferências de indução. Esses algoritmos são descritos a seguir:

**C4.5** Como já mencionando, o sistema C4.5 (Quinlan, 1993) é um membro da família de algoritmos dividir-para-conquistar. Modificamos a árvore induzida para estimar probabilidades em cada folha utilizando a correção de Laplace, e essas probabilidades foram utilizadas para ordenar os casos. Utilizamos tanto a árvore não podada quanto a podada, treinada com parâmetros padrão.

**CN2** Assim como com o C4.5, utilizamos a correção de Laplace para estimar probabilidades e ordenar casos, mas nesse caso as probabilidades foram calculadas para todas as regras que cobrem um dado exemplo.

**Redes neurais to tipo MLP e RBF** Utilizamos uma rede neural do tipo perceptron multi-camada (MLP) e também do tipo função de base radial (RBF), treinadas utilizado o algoritmo de *backpropagation*, com camadas de entradas e oculta com o número de nós equivalente ao número de atributos no conjunto de dados. O coeficiente de momento foi ajustado para 0,2, a taxa de aprendizado para 0,3 e o número máximo de épocas foi ajustado para 500. O valor contínuo predito pelas redes neurais foi utilizado para ordenar os casos.

**SVM** Utilizamos a implementação $SVM^{light}$ (Joachims, 1999) das máquinas de vetores de suporte (SVM). Utilizamos tanto a versão para classificação — SVM(C) — e a adaptada para ordenar casos utilizando pares de exemplos (Joachims, 2002). Em ambos os casos, as SVMs foram treinadas utilizando *kernel* linear e parâmetros padrão. A distância até às margens dos vetores de suporte foi utilizada para ordenar os casos.

**Naïve Bayes** Também utilizamos o Naïve Bayes — NB. As probabilidades *a posteriori* fornecidas pelo NB foram utilizadas para ordenar os casos.

Executamos os experimentos 10 vezes e a cada execução foi utilizada a validação cruzada com 10 partições (10×10-cv). Para cada partição da validação cruzada, 90% dos exemplos foram utilizados para treinamento e o restante foi utilizado para a ordenação. O experimento é pareado, *i.e.*,

para todos os algoritmos foram fornecidos os mesmos conjunto de treinamento e teste. Na Tabela 8.3 são apresentados, para cada conjunto de dados e para cada *ranking* base, assim como para o BORDARANK, o valor médio da AUC de todas as execuções. Desvios padrão são mostrados entre parêntesis. Para uma melhor visualização, a célula com o valor mais alto da AUC obtido para cada um dos conjuntos de dados está destacada por uma cor cinza-escuro, e a segunda melhor AUC com a cor cinza-claro. Para testar se existem diferenças entre os algoritmos, executamos o teste de Friedman (Demšar, 2006), que é uma versão equivalente não paramétrica à ANOVA, com a hipótese nula que o desempenho de todos os algoritmos, medidos em termos da AUC são comparáveis. Dessa maneira, testamos a seguinte hipótese:

$$\begin{cases} H_0: & \text{AUC todos algoritmos são comparáveis} \\ H_1: & \text{AUC todos algoritmos não são comparáveis} \end{cases}$$

Segundo a estatística de Friedman, a hipótese nula $H_0$ pode ser descartada com 95% de confiança.

Como a hipótese nula foi rejeitada, podemos prosseguir com um teste para detectar se as diferenças entre os algoritmos são significativas. Executamos o teste de múltiplas comparações com um controle de Bonferroni-Dunn (Demšar, 2006), utilizando BORDARANK como controle. Em outras palavras, testamos a seguinte hipótese

$$\begin{cases} H_0: & AUC_{\text{BORDARANK}} = AUC_{\text{outros métodos}} \\ H_1: & AUC_{\text{BORDARANK}} \neq AUC_{\text{outros métodos}} \end{cases}$$

Com 95% de confiança, podemos rejeitar a hipótese nula de que BORDARANK tem um desempenho comparável aos seguintes algoritmos: C4.5, C4.5(NP), NB, CN2, SVM(C) e RBF. Isso é equivalente a dizer que BORDARANK é significativamente melhor que esses *rankings* base com 95% de confiança. Mesmo que o teste não tenha detectado diferenças significativas quando BORDARANK é comparado à SVM(R) e MLP, as diferenças entre os resultados de BORDARANK, SVM(R) e MLP são próximas à diferença crítica entre os resultados para a rejeição da hipótese nula (BORDARANK, MLP e SVM(R) tiveram um número de pontos 2.14, 3.69 e 4.19 respectivamente. A diferença entre BORDARANK e MLP é de 1.50 e entre BORDARANK e SVM(R) é de 2.05. A hipótese nula pode ser rejeitada com 95% de confiança se a diferença entre o número de pontos entre os dois algoritmos é maior que 2.28).

Os resultados apresentados até aqui nos permitem afirmar que BORDARANK melhora sensivelmente com relação aos *rankings* base. Mesmo que

| # | C4.5 | C4.5(NP) | NB | CN2 | SVM(R) | SVM(C) | MLP | RBF | BORDARANK |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 97.9(1.6) | 98.3(1.5) | 97.4(1.6) | 99.1(0.7) | 99.5(0.6) | 99.5(0.5) | 98.4(1.5) | 84.7(19.5) | 99.4(0.7) |
| 2 | 67.7(8.1) | 68.0(8.0) | 64.7(9.9) | 67.7(8.3) | 71.1(9.0) | 69.7(10.9) | 71.9(8.6) | 71.2(9.9) | 76.0(8.3) |
| 3 | 83.8(7.9) | 85.1(7.3) | 90.0(5.8) | 81.3(7.3) | 91.1(5.7) | 74.4(10.6) | 88.6(6.5) | 81.3(8.7) | 90.5(5.7) |
| 4 | 85.7(11.8) | 86.9(10.8) | 87.3(10.6) | 48.6(1.5) | 90.7(10.9) | 93.2(7.4) | 89.4(9.5) | 87.3(18.7) | 92.7(7.4) |
| 5 | 50.0(0.0) | 69.8(23.6) | 57.2(20.3) | 65.5(21.7) | 53.2(23.1) | 42.5(27.8) | 61.5(23.6) | 53.8(17.7) | 65.0(24.6) |
| 6 | 51.1(3.4) | 72.7(5.6) | 76.0(5.3) | 60.7(6.6) | 75.3(5.3) | 60.9(8.8) | 65.2(7.2) | 62.8(11.0) | 72.2(5.9) |
| 7 | 70.1(5.3) | 69.8(5.3) | 78.1(5.2) | 73.7(5.0) | 62.1(7.5) | 62.5(5.3) | 71.9(5.2) | 57.0(6.8) | 77.4(5.1) |
| 8 | 81.3(19.6) | 81.5(19.3) | 71.3(20.1) | 73.0(14.1) | 65.1(15.5) | 54.2(20.5) | 83.1(18.1) | 81.3(13.7) | 84.8(12.7) |
| 9 | 58.5(8.4) | 61.7(9.4) | 64.8(10.4) | 56.7(10.7) | 67.8(9.9) | 70.1(10.1) | 68.0(10.7) | 66.0(12.6) | 68.5(10.2) |
| 10 | 83.0(7.0) | 84.4(6.4) | 90.0(5.9) | 72.6(8.9) | 89.6(5.6) | 73.8(10.1) | 86.6(5.9) | 86.0(7.1) | 89.8(5.3) |
| 11 | 93.1(5.0) | 93.5(4.9) | 90.2(5.7) | 84.9(8.6) | 88.9(6.6) | 90.6(5.7) | 94.0(4.3) | 73.7(13.8) | 96.3(3.0) |
| 12 | 99.9(0.2) | 99.9(0.1) | 95.2(1.1) | 99.9(0.2) | 99.6(0.2) | 98.7(0.5) | 99.8(0.2) | 50.0(0.0) | 99.9(0.1) |
| 13 | 94.2(8.8) | 94.7(8.6) | 100.0(0.0) | 94.8(12.3) | 99.9(0.5) | 91.8(12.6) | 99.9(0.4) | 93.7(16.1) | 99.7(0.9) |
| 14 | 76.5(5.4) | 77.7(5.1) | 81.1(4.4) | 78.4(4.9) | 76.4(4.9) | 76.4(5.3) | 74.4(5.6) | 76.2(6.9) | 82.4(4.2) |
| 15 | 78.3(11.8) | 78.2(11.8) | 77.7(9.9) | 50.6(1.8) | 83.4(8.4) | 82.0(9.1) | 90.0(6.7) | 85.6(8.4) | 89.8(7.1) |
| 16 | 92.0(2.7) | 93.4(2.5) | 74.4(4.9) | 100.0(0.0) | 99.7(0.2) | 98.8(1.1) | 99.4(0.8) | 59.1(7.8) | 99.1(0.6) |
| 17 | 95.8(2.9) | 97.0(1.5) | 91.2(4.0) | 93.8(3.9) | 96.3(1.5) | 96.1(1.6) | 94.2(2.3) | 88.0(5.5) | 97.8(1.2) |
| 18 | 97.5(1.5) | 97.5(1.5) | 81.1(4.8) | 96.3(1.9) | 98.7(1.5) | 90.7(4.0) | 99.8(0.2) | 95.3(4.5) | 98.8(0.9) |

**Tabela 8.3:** Valores da AUC dos *rankings* base e do BORDARANK. Células destacadas em cinza escuro indicam o melhor valor da AUC. Células destacas em cinza escuro o segundo melhor valor da AUC.

o teste de Bonferroni-Dunn não tenha mostrado diferenças significativas com relação a SVM(R) e MLP, uma inspeção da Tabela 8.3 mostra que a coluna correspondente ao BORDARANK concentra a maior parte dos melhores ou segundo melhores valores da AUC. Além disso, em dois dos quatro conjuntos nos quais BORDARANK não obteve o melhor ou o segundo melhor resultado em termos da AUC (#13 New-thyroid e #16 tic-tac-toe), existe um algoritmo com AUC igual a 100%, e não há como melhorar com relação a esse valor. No mais, para os outros dois conjuntos de dados nos quais BORDARANK não teve o melhor ou o segundo melhor valor da AUC (#5 flag e #6 fare), a maioria dos algoritmos obteve valores de AUC menores que 65%, e somente alguns algoritmos obtiveram AUC maior do que esse valor. Como BORDARANK é uma combinação desses *rankings* base, os *rankings* com baixo desempenho não estão contribuindo para a melhoria de BORDARANK.

No segundo estágio do experimento, comparamos BORDARANK com vários métodos para combinar o valor contínuo predito pelos algoritmos de aprendizado utilizados como *ranking* base no primeiro estágio dos experimentos. Para prevenir problemas de escala, todos os valores foram ajustados entre zero e um. Os métodos utilizados neste estágio do experimento são discutidos em (Kuncheva, 2004, Capítulo 5), e são brevemente descritos a seguir.

**Média dos valores preditos** Calcula-se a média de todos os valores preditos para um dado caso, e os casos são reordenados com relação a essa média. Esse método é similar à regra da soma, discutida na Seção 8.3.

**Produto dos valores preditos** Calcula-se o produto de todos os valores preditos para um dado caso, e os casos são reordenados com relação a esse produto. Esse método é similar à regra do produto, discutida na Seção 8.3.

**Média aparada dos valores preditos** Os dois melhores e os dois piores valores da AUC para cada caso são removidos e a média dos quatro valores restantes da AUC é calculada. A média aparada tem o objetivo de prevenir problemas causados pela distorção de valores muito altos ou muito baixos que podem afetar a média aritmética.

**Combinação utilizando uma rede neural tipo MLP** Utilizamos uma rede neural do tipo MLP como um meta-algoritmo — Meta(MLP) — para aprender uma função que combine os valores individuais preditos para os casos do conjunto de treinamento. A rede treinada é então utilizada para combinar os valores preditos para os casos do conjunto de teste,

e os casos são então ordenados utilizando a função combinada. A topologia da rede inclui duas camadas ocultas com 5 neurônios cada. Ela foi treinada utilizando o algoritmo de *backpropagation* padrão, o coeficiente de momento foi ajustado para 0,2, a taxa de aprendizado para 0,3 e o número máximo de épocas foi ajustado para 1000

**Combinação utilizando Naïve Bayes** Como no item anterior, mas utilizando o algoritmo Naïve Bayes — Meta(NB) — como meta-algoritmo.

**Combinação utilizando SVM** Como no item anterior, mas utilizando o algoritmo SVM — Meta(SVM) — como meta algoritmo. SVMs foram treinadas utilizando kernel linear e com parâmetros padrão.

Na Tabela 8.4 são apresentados os valores médios da AUC de todas as execuções para esses seis métodos de combinação da predição dos algoritmos base. Para facilitar a comparação, os resultados do BordaRank estão repetidos. Assim como mostrado na Tabela 8.3, as células com melhores valores de AUC estão destacadas utilizando-se uma cor cinza-escuro. De maneira similar à primeira série de experimentos, executamos o teste de Friedman para testar a hipótese nula de que o desempenho dos métodos de combinação das predições são comparáveis em termos da AUC. Dessa maneira, testamos a seguinte hipótese:

$$\begin{cases} H_0: & \text{AUC todos métodos de combinação são comparáveis} \\ H_1: & \text{AUC todos métodos de combinação não são comparáveis} \end{cases}$$

Segundo a estatística de Friedman, a hipótese nula $H_0$ pode ser descartada com 95% de confiança. Como a hipótese nula foi rejeitada, podemos executar novos testes para verificar como BordaRank é comparável com os métodos de combinação de predições. Também executamos o teste de Bonferroni-Dunn, com BordaRank como controle. Em outras palavras, testamos a seguinte hipótese:

$$\begin{cases} H_0: & AUC_{\text{BordaRank}} = AUC_{\text{outros métodos de combinação}} \\ H_1: & AUC_{\text{BordaRank}} \neq AUC_{\text{outros métodos de combinação}} \end{cases}$$

O teste mostra que BordaRank é estatisticamente melhor, com um grau de confiança de 95%, do que o produto dos valores preditos e a combinação construída com a rede neural MLP (Meta(MLP)) e Naïve Bayes (Meta(NB)). Nenhuma diferença significativa foi encontrada com relação à média e média aparada dos valores preditos e a combinação construída pelo SVM (Meta (SVM)).

| # | BORDARANK | Average | Produto | Aparada | Meta(MLP) | Meta(NB) | Meta(SVM) |
|---|---|---|---|---|---|---|---|
| 1 | 99.4(0.7) | 99.4(0.6) | 95.5(7.1) | 99.4(0.7) | 97.1(2.3) | 91.7(11.9) | 98.4(1.8) |
| 2 | 76.0(8.3) | 76.2(8.6) | 74.1(8.2) | 76.0(8.7) | 67.7(8.1) | 70.2(10.6) | 73.6(8.5) |
| 3 | 90.5(5.7) | 90.5(5.7) | 86.2(7.7) | 90.5(5.7) | 78.9(8.8) | 82.6(7.2) | 88.0(7.0) |
| 4 | 92.7(7.4) | 92.3(9.3) | 87.5(14.1) | 92.9(8.3) | 81.7(12.5) | 63.3(19.1) | 79.7(15.4) |
| 5 | 65.0(24.6) | 61.9(26.0) | 54.3(14.4) | 64.2(24.6) | 64.1(23.3) | 50.0(0.0) | 64.4(24.4) |
| 6 | 72.2(5.9) | 74.2(6.2) | 73.1(7.3) | 74.0(6.0) | 61.3(8.3) | 51.8(5.8) | 61.3(7.4) |
| 7 | 77.4(5.1) | 77.3(5.4) | 62.7(7.4) | 77.8(5.1) | 67.0(5.5) | 68.0(4.7) | 73.9(5.1) |
| 8 | 84.8(12.7) | 82.5(17.1) | 84.8(17.2) | 82.8(17.5) | 73.6(20.3) | 59.7(16.6) | 88.6(14.1) |
| 9 | 68.5(10.2) | 68.6(10.4) | 66.7(12.0) | 69.2(10.3) | 55.3(11.2) | 58.0(10.9) | 60.7(10.5) |
| 10 | 89.8(5.3) | 90.0(5.5) | 86.9(6.3) | 89.9(5.4) | 79.2(7.9) | 80.0(6.5) | 86.2(6.3) |
| 11 | 96.3(3.0) | 96.4(3.3) | 81.2(8.9) | 95.7(3.7) | 94.0(4.0) | 66.5(21.4) | 95.4(4.2) |
| 12 | 99.9(0.1) | 99.9(0.1) | 77.0(25.0) | 99.9(0.1) | 99.7(0.3) | 50.0(0.0) | 99.9(0.1) |
| 13 | 99.7(0.9) | 99.9(0.3) | 98.1(7.2) | 99.9(0.3) | 99.8(0.8) | 72.5(25.1) | 99.9(0.6) |
| 14 | 82.4(4.2) | 82.8(4.1) | 74.9(6.6) | 82.7(4.3) | 71.7(6.2) | 73.3(9.9) | 76.9(5.2) |
| 15 | 89.8(7.1) | 89.6(7.6) | 86.2(8.3) | 89.6(7.7) | 86.9(8.4) | 55.4(12.4) | 89.7(7.6) |
| 16 | 99.1(0.6) | 99.8(0.3) | 80.3(11.6) | 99.8(0.2) | 99.9(0.3) | 99.7(0.8) | 100.0(0.0) |
| 17 | 97.8(1.2) | 97.6(1.3) | 93.9(2.7) | 97.8(1.2) | 92.6(3.1) | 88.6(12.4) | 95.7(2.2) |
| 18 | 98.8(0.9) | 99.3(0.6) | 97.8(4.3) | 99.3(0.6) | 99.0(1.5) | 93.2(13.6) | 99.7(0.3) |

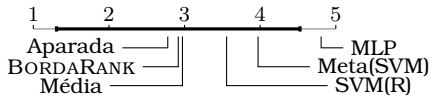
**Tabela 8.4:** Valores da AUC para o BORDARANK e os métodos de combinação da predição contínua dos modelos base. Célula destacadas em cinza escuro indicam o melhor valor da AUC para cada conjunto de dados.

Os resultados mostram que BORDARANK é competitivo com a média, média aparada e a combinação construída utilizando SVM. Eles também mostram que BORDARANK, assim como média, média aparada e Meta(SVM) são melhores que o produto das predições, do Meta(NB) e Meta(MLP). Esses resultados são muito interessantes uma vez que BORDARANK somente utiliza a ordenação dos casos, enquanto que os outros algoritmos utilizam as predições numéricas individuais produzidas pelos *rankings* base. Em outras palavras, BORDARANK pode ser utilizado em casos que somente *rankings* de casos estejam disponíveis, tal como em aplicações em que somente preferências de usuários estejam disponíveis, mas com um desempenho comparável com abordagens mais sofisticadas que utilizam valores contínuos.

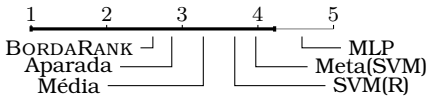
Para concluir a avaliação experimental, no terceiro estágio da avaliação, para simular o uso do *ranking* combinado para derivar modelos de classificação, derivamos modelos que predizem os exemplos que aparecem nas 5%, 10%, 20% e 30% primeiras posições dos respectivos *rankings*, e avaliamos precisão (número de exemplos positivos nas $n$% primeiras posições dividido pelo número total de exemplos nas $n$% primeiras posições) e sensitividade (número de exemplos positivos nas $n$% primeiras posições dividido pelo número total de exemplos positivos)[2]. Para uma melhor visualização, apenas a comparação dos resultados (também utilizando o teste Bonferroni-Dunn com 95% de confiança) é mostrado graficamente na Figura 8.1.

Além disso, incluímos apenas abordagens ques não tiveram uma melhora significativa em termos da AUC com relação ao BORDARANK (média e média aparada dos valores preditos, Meta(SVM), SVM(R) e MLP). Em cada gráfico dessa figura, a linha mais grossa marca o intervalo de uma diferença crítica (como descrito em (Demšar, 2006)), à direita e à esquerda, quando comparado com BORDARANK. Valores à esquerda (direta) dessa linha são significativamente melhores (piores) do que BORDARANK. Mesmo que os gráficos não mostrem muitas diferenças significativas, podemos identificar dois grupos de algoritmos: a média e a média aparada, juntamente com BORDARANK, quase sempre aparecem juntos, alternando os melhores resultados. Já SVM(R), MLP e Meta(SVM) formam outro grupo, alternando os piores resultados. Essa é uma outra evidência que BORDARANK produz resultados que são comparáveis à média e a média aparada, mesmo que BORDARANK utilize somente as posições no *ranking* para construir o *ensemble* final.

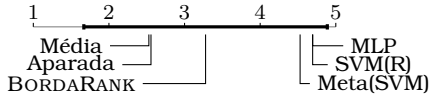
[2]Estamos simulando uma tarefa similar à recuperação de informação, na qual estamos geralmente interessados em quão bem classificamos exemplos da classe positiva. Se estivéssemos interessados na derivação de modelos de classificação para ambas as classes, uma abordagem mais aconselhável é classificar como positivos uma porcentagem igual à proporção natural de exemplos na população, e avaliar a taxa de erro de classificação.



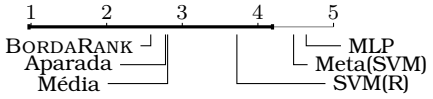

(c) Precisão com 5% dos exemplos positivos



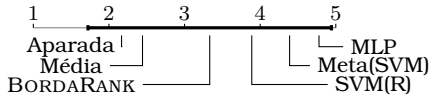

(d) Sensitividade com 5% dos exemplos positivos



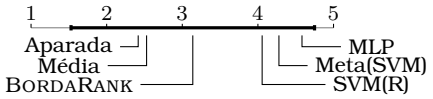

(e) Precisão com 10% dos exemplos positivos



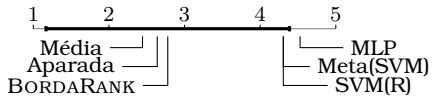

(f) Sensitividade com10% dos exemplos positivos



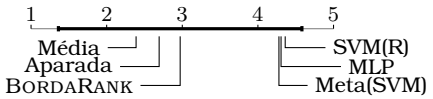

(g) Precisão com 20% dos exemplos positivos




(h) Sensitividade com 20% dos exemplos positivos




(i) Precisão com 30% dos exemplos positivos




(j) Sensitividade com 30% dos exemplos positivos

**Figura 8.1:** Comparação da precisão e sensitividade com 5%, 10%, 20% e 30% dos exemplos classificados como positivos.

## 8.7. Considerações finais

O problema de ordenação de exemplos vêm recebendo uma grande atenção da comunidade de AM nos últimos anos. Além disso, a construção de *ensembles* é um método muito aplicado e com bons resultados para melhorar o desempenho de modelos de classificação. Entretanto, pouco trabalho tem sido feito na combinação desses dois métodos.

Neste capítulo apresentamos um método simples, inspirado na votação por preferências múltiplas com ordenação borda count. Os resultados experimentais mostraram um ganho de desempenho com base nos *rankings* individuais que compõem o *ensemble*. Além disso, BORDARANK é competitivo com vários outros métodos de combinar os valores contínuos preditos pelos modelos induzidos pelos algoritmos de aprendizado. Como BORDARANK precisa apenas da ordenação dos exemplos e não desse valor contínuo para fazer a combinação, ele tem uma gama maior de possíveis aplicações.

Capítulo 9

# Conclusão

Neste capítulo é apresentada a conclusão deste trabalho. Na Seção 9.1 é feito um paralelo entre os objetivos desta tese e os resultados obtidos. Na Seção 9.2 são discutidas as limitações dos resultados apresentados neste trabalho. Finalmente, na Seção 9.3 são apresentadas as direções de trabalhos futuros.

## 9.1. Resumo dos objetivos e principais resultados

A intersecção entre as áreas de aprendizado de máquina e mineração de dados tem-se mostrado uma rica fonte de pesquisas. Por um lado, mineração de dados tem uma grande necessidade de novas abordagens para resolver problemas práticos do mundo real. Do outro lado, aprendizado de máquina provê o núcleo da maioria das práticas de mineração de dados.

Nesta tese, exploramos algumas das limitações ou restrições de algoritmos de aprendizado que vieram à tona com as suas aplicações em mineração de dados com o intuito de aliviar essas limitações propondo novas abordagens, métodos e algoritmos dentro da área de aprendizado de máquina. A seguir, reproduzimos os objetivos apresentados na introdução desta tese e contextualizamos os resultados apresentados ao longo dela dentro desses objetivos.

1. *Podemos explorar alguma representação alternativa para a indução de modelos? Como incorporar essa representação em um algoritmo de aprendizado? Preferencialmente, representações que sejam mais facilmente entendíveis por seres humanos (especilistas ou não).*

   Com relação a esse objetivo, propomos um método para a geração de execeções a partir de regras gerais. Com essas exceções, contextualizamos o conhecimento adquirido pelo par (regra geral, exceção). A contextualização provida pelo par regra geral e exceção é uma maneira muito natural de humanos expressarem conhecimento. Os resultados obtidos foram publicados em (Prati et al., 2004e,d, 2003c,d,e, 2004c).

2. *Podemos explorar algum método de busca alternativo na geração de conjuntos de regras? Como o desempenho desse método se compara com outros métodos?*

   Para esse objetivo, propomos o algoritmo de seleção de regras ROCCER, que utiliza a região convexa da curva ROC para selecionar regras. Os resultados mostraram um desempenho comparável a algoritmos clássicos de indução de regras, mas com um número de regras muito menor do que aqueles apresentados por esses algoritmos clássicos. Nossa abordagem é diferente da abordagem tradicional de cobertura de conjunto pois o ROCCER provê um mecanismo natural de *backtracking*, o que possibilita a exclusão de uma regra previamente inserida à lista de decisão ou re-ordenação dessas regras. Os resultados obtidos foram publicados em (Prati & Flach, 2004, 2005).

3. *Conjuntos de dados com classes desbalanceadas podem piorar o desempenho de algoritmos de aprendizado? Como? Em que condições? Podemos compensar o problema causado pelas classes desbalanceadas? Como?*

   Esta tese apresenta várias contribuições com respeito a esses objetivos. Primeiramente, constatamos experimentalmente a nossa hipótese de que o desbalanceamento entre as classes não é sempre um problema, mas é um agravante quando relacionado a outros, tais como uma grande sobreposição de classes. Também constatamos experimentalmente que, dentro do contexto de árvores de decisão, poda geralmente é prejudicial ao desempenho de algoritmos de aprendizado na presença de desbalanceamento entre as classes. Também constatamos que métodos de *over-sampling* têm, em geral, um desempenho melhor do que métodos de *under-sampling* no balanceamento artificial de conjuntos de dados. Isso se deve ao fato que *under-sampling* pode descartar exemplos potencialmente úteis para o aprendizado. Uma

outra constatação experimental é que métodos de balanceamento artificial de conjuntos de dados tendem a piorar o problema de pequenos disjuntos. Um dos métodos por nós proposto contribui para a diminuição desse problema, apesar da concentração de erro nos pequenos disjuntos ainda ser alta. Finalmente, verificamos que a proporção em que rotulamos exemplos no algoritmo de aprendizado semi-supervisionado Co-training é um parâmetro de fundamental importância, fato muitas vezes ignorado pela comunidade. Os resultados obtidos foram publicados em (Prati et al., 2004a; Batista et al., 2005, 2004; Prati et al., 2003a, 2004b; Matsubara et al., 2006)

4. *Como combinar diversos rankings de tal maneira a construir um ranking final com melhor desempenho que os rankings originais para o problema de ordenação binária? Como o desempenho desse ranking se compara com outras abordagens? Como utilizar rankings para derivar modelos de classificação?*

   Para esse objetivo, propomos um algoritmo para a combinação de vários *rankings* em um *ranking* final, abordagem conhecida como *ensemble* de *rankings*. Nosso algoritmo, denominado BordaRank, calcula a média dos *rankings* e depois reordena os casos com base nessa média. Os resultados mostram uma melhora de desempenho, em termos da AUC, com relação aos *rankings* individuais, além de um desempenho comparável com métodos que utilizam mais informação (os valores numérios das predições dos modelos gerados pelos algoritmos de AM) para o mesmo fim. Uma vantagem clara é que, uma vez que nossa abordagem utiliza apenas as posições das ordenações dos exemplos, ela é mais geral e aplicável em situações em que somente vários *rankings* de casos estejam disponíveis.

## 9.2. Limitações

Nesta seção destacamos as principais limitações das soluções propostas relacionadas aos objetivos desta tese.

1. A abordagem para encontrar exceções foi validada com um estudo de caso. Entretanto, seria interessante avaliar a proposta com um número maior de conjuntos de dados. Porém, um problema quanto à essa avaliação é que ela compromete um aspecto importante da avaliação, àquele relacionado à interpretação das regras que compõem o modelo, uma vez que uma análise desse tipo implica em executar o algoritmo várias vezes para um mesmo conjunto de dados (validação

cruzada), gerando um grande número de modelos, o que dificulta a avaliação "semântica" desses modelos.

2. Uma das desvantagens do algoritmo ROCCER é que ele gera listas de decisão. Como dito no Capítulo 5, listas de decisão são de difícil compreensão, pois as regras são válidas somente no contexto das anteriores. Uma outra desvatagem é o tempo de execução que, em alguns casos, é alto. Esse fato pode ser contornado pela manipulação dos parâmetros suporte e confiança do algoritmo *Apriori*. Uma boa heurística para ajustar esses parâmetros, no entanto, seria desejável.

3. Quanto aos experimentos reportados no Capítulo 6, uma das principais limitações está relacionada ao uso de apenas um algoritmo de aprendizado para a realização dos experimentos. Além disso, as análises foram baseadas somente em termos da AUC. Para se ter um melhor entendimento do desempenho dos métodos de balanceamento artificial de conjuntos de dados, seria interessante analisar o comportamento das curvas ROC em todo o seu espectro.

4. Quanto ao algoritmo BORDARANK, o método descrito é na realidade um procedimento geral, que pode ser aplicado a qualquer conjunto de funções que mapeiem os casos a um conjunto ordenado de casos. No experimento desenvolvido no Capítulo 8, em particular, para criar essas funções utilizamos diferentes algoritmos de aprendizado de máquina que predizem um valor contínuo, o qual pode ser entendido como uma estimativa das chances de um caso pertencer à uma classe. Essa escolha não é, necessariamente, a melhor. Uma outra possível escolha seria, por exemplo, utilizar variações dos parâmetros de um mesmo algoritmo. Também seria interessante uma alternativa para descartar modelos muito ruins do *ensemble* final. Finalmente, uma comparação direta com algoritmos desenvolvidos para o mesmo fim (como o RankBoost (Freund et al., 2003)) é desejável. Essa comparação será realizada em trabalhos futuros, como descrito a seguir.

## 9.3. Trabalhos futuros

Várias extensões podem ser exploradas relaciondas aos trabalhos realizados. Uma das possíveis extensões é explorar as interconexões entre os diversos temas tratados nesta tese. Seria interessante, por exemplo, unificar as duas novas abordagens para a geração de regras. Uma concavidade

na curva ROC pode representar um exceção e esse fato poderia ser explorado para unificar as duas abordagens.

Um outro trabalho interessante é verificar o comportamento dessas abordagens no problema de classes desbalanceadas. Exceções, por exemplo, poderiam ser utilizadas para melhorar a classificação das classes minoritárias. Ou, ainda, fornecer ao ROCCER um número maior de regras da classe minoritária (relaxando o critério de suporte do *Apriori* somente para essa classe) e verificar o seu desempenho em domínios com esse problema. Também seria interessante averiguar o desempenho do BORDARANK com os problemas de classes desbalanceadas. Nesse caso, o objetivo seria ordenar os exemplos da classe minoritária à frente dos exemplos da classe majoritária.

Quanto ao ROCCER, como ele é um procedimento genérico de seleção de regras, ele pode ser explorado de várias maneiras. Seria interessante, por exemplo, verificar o comportamento do ROCCER para combinar regras de diferentes algoritmos de aprendizado, em uma abordagem parecida com a descrita em (Baranauskas & Monard, 2003). Ou, ainda, como uma estratégia de poda de regras, na qual um conjunto inicial de regras induzidas por outro algoritmo de indução de regras é utilizado como entrada para o ROCCER. Uma outra aplicação interessante do ROCCER está relacionada ao aprendizado em fluxo contínuo de dados. Como ROCCER provê um mecanismo de *backtracking*, ele pode ser facilmente adaptado para condições de fluxo contínuo de dados. Nesse caso, seria utilizado como entrada a ser fornecida ao algoritmo uma adaptação de *Apriori* para fluxos contínuos de dados (Manku & Motwani, 2002). Finalmente seria interessante investigar maneiras de transformar a lista de decisão em um conjunto de regras, com o objetivo de propiciar um melhor entendimento das regras geradas pelo ROCCER.

Um outro trabalho futuro interessante é utilizar diferentes algoritmos de aprendizado e repetir as experiências com classes desbalanceadas reportadas no Capítulo 6. Também seria interessante utilizar, ao invés dos métodos de amostragem, custos para modificar os pesos relativos de cada classe. Dessa maneira, em um método de *over-sampling*, ao invés de duplicar os exemplos da classe minoritária em $n$ vezes, aumentaríamos o custo de cometer um erro nessa classe em $n$ unidades. Uma possível limitação dessa abordagem é que muito dos algoritmos atuais de aprendizado não são preparados para lidar com custos. Ainda, os que são preparados tratam uniformemente os custos de cada classe, e não é possível utilizar métodos heurísticos que aumentariam os custos de apenas alguns dos exemplos, como os métodos heurísticos de amostragem fazem.

Finalmente, quanto ao BORDARANK, gostaríamos de compará-lo com o RankBoost. Essa comparação depende de uma implementação desse algoritmo (já que a versão original não foi disponibilizada pelos autores). Uma possível extensão desse algoritmo seria primeiramente criar vários "pequenos" *ensembles* com todos os possíveis pares de *rankings*. Após, também criaríamos *ensembles* combinando os rankings três a três, e assim sucessivamente até a combinação de $n-1$ *rankings*, construindo o *ranking* final a partir dos *rankings* originais e dos "pequenos" *ensembles* construídos. Essa abordagem é computacionalmente simples e econômica, e poderia resultar em um ganho de desempenho.

Além disso, pretendemos continuar investigando o uso de *rankings* para avaliar algoritmos de aprendizado segundo multiplos critérios, como recentemente por nós proposto em (Lee et al., 2006), explorando as vantagens de considerar apenas as ordenações das medidas obtidas, o que nos permite considerar diferentes medidas em diferentes escalas. Essas medidas podem então ser combinadas em um *ranking* final. Aliás, vale salientar que ainda que *rankings* seja um método bem conhecido na área de estatística não-paramétrica, ele tem sido pouco explorado na área de aprendizado de máquina.

# Referências Bibliográficas


Agrawal, R., Imielinski, T., & Swami, A. N. (1993). Mining association rules between sets of items in large databases. In *ACM SIGMOD International Conference on Management of Data*, pag. 207–216, Washington, D.C. Citado na página 69.

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1):37–66. Citado na página 20.

Baranauskas, J. A. & Monard, M. C. (2003). Combining symbolic classifiers from multiple inducers. *Knowl.-Based Syst.*, 16(3):129–136. Citado na página 153.

Batista, G. E. A. P. A. (2003). Pré-processamento de Dados em Aprendizado de Máquina Supervisionado. Tese de Doutorado, ICMC-USP, `http://www.teses.usp.br/teses/disponiveis/55/55134/tde-06102003-160219/`. Citado na página 93.

Batista, G. E. A. P. A., Milaré, C. R., Prati, R. C., & Monard, M. C. (2006). A comparison of methods for rule subset selection applied to associative classification. In *Eighth Argentine Symposium on Artificial Intelligence (ASAI 2006)*. aceito. Citado na página 8.

Batista, G. E. A. P. A. & Monard, M. C. (2003a). An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5–6):519–533. Citado na página 5.

Batista, G. E. A. P. A. & Monard, M. C. (2003b). Descrição da arquitetura e do projeto do ambiente computacional DISCOVER LEARNING ENVIRONMENT — DLE. Technical Report 187, ICMC-USP. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_187.pdf`. Citado na página 93.

Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations*, 6(1):20–29. Citado nas páginas 8, 52, 90, 95, 106, 125, e 151.

Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C. (2005). Balancing strategies and class overlapping. In *International Symposium on Intelligent Data Analysis (IDA'2005)*, volume 3646 of *Lecture Notes in Computer Science*, pag. 24–35, Madrid (Spain). Springer. Citado nas páginas 8, 99, e 151.

Bauer, E. & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139. Citado na página 109.

Bernardini, F. C., Monard, M. C., & Prati, R. C. (2005). Constructing ensembles of symbolic classifiers. In *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*, pag. 315–320. IEEE Computer Press. Convidado para possível publicação no International Journal on Hybrid Intelligent Systems (IJHIS). Citado na página 9.

Bernardini, F. C., Monard, M. C., & Prati, R. C. (2006). Constructing ensembles of symbolic classifiers. *International Journal on Hybrid Intelligent Systems (IJHIS)*. special issue on "Ensemble Approaches", relativo ao HIS'05, submetido. Citado na página 9.

Blum, A., Kalai, A., & Wasserman, H. (2003). Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519. Citado na página 5.

Blum, A. & Mitchell, T. (1998). Combining labeled and unlabeled data with Co-training. In *Proceedings 11th Annu. Conf. on Comput. Learning Theory*, pag. 92–100. ACM Press. Citado na página 127.

Borda, J.-C. (1781). *Memoire sur les Elections au Scrutin*. Historie de l'Academie Royale des Sciences, Paris. Citado nas páginas 134 e 136.

Borgelt, C. & Kruse, R. (2002). Induction of association rules: A priori implementation. In *15th Conf. on Computational Statistics*, pag. 395–400, Heidelberg, Germany. Physica Verlag. Citado na página 81.

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7). Citado na página 50.

Braga, A. P., Ludimir, T. B., & Carvalho, A. C. P. L. F. (2003). Redes neurais artificiais. chapter 6, pag. 141–168. Manole, Barueri, SP, Brasil. Citado na página 20.

Bratko, I. (2001). *Prolog Programming for Artificial Intelligence*. Addison-Wesley, 3 edition. Citado na página 24.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140. Citado nas páginas 134 e 138.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*. Wadsworth & Books, Pacific Grove, CA. Citado nas páginas 18, 19, 42, 43, e 89.

Caruana, R., Baluja, S., & Mitchell, T. M. (1995). Using the future to sort out the present. In *Neural Information Processing Systems (NIPS'1995)*, pag. 959–965, Denver, USA. MIT Press. Citado na página 135.

Chawla, N., Japkowicz, N., & Kolcz, A., editors (2003). *ICML'2003 Workshop on Learning from Imbalanced Data Sets (II)*. Proceedings available at `http://www.site.uottawa.ca/~nat/Workshop2003/workshop2003.html`. Citado nas páginas 88 e 160.

Chawla, N., Japkowicz, N., & Kolcz, A. (2004). Special issue on learning from imbalanced datasets. *SIGKDD Exploration*, 6(1). Citado na página 88.

Chawla, N. V. (2003). C4.5 and imbalanced data sets: Investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In *Workshop on Learning from Imbalanced Data Sets II*. Citado na página 109.

Chawla, N. V. (2005). Data mining for imbalanced datasets: An overview. In Maimon, O. & Rokach, L., editors, *The Data Mining and Knowledge Discovery Handbook*, pag. 853–867. Springer. Citado na página 88.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Jounal of Artificial Intelligence Research*, 16:321–357. Citado na página 95.

Clark, P. & Boswell, R. (1991). Rule induction with CN2: Some recent improvements. In *Proceedings of the 5th European Conf. on Machine Learning*, volume 482 of *Lecture Notes in Artificial Intelligence*, pag. 151–163. Springer. Citado nas páginas 19, 32, 40, e 80.

Clark, P. & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, 3(4):261–283. Citado nas páginas 19, 32, e 79.

Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of the 12th Int. Conf. on Machine Learning*, pag. 115–123. Citado na página 80.

Cohen, W. W., Schapire, R. E., & Singer, Y. (1999). Learning to order things. *J. Artif. Intell. Res. (JAIR)*, 10:243–270. Citado nas páginas 135 e 137.

Cohen, W. W. & Singer, Y. (1999). A simple, fast and effective rule learner. In *Proceedings of 16th National Conf. on Artificial Intelligence (AAAI-99)*, pag. 335–342. AAAI/MIT Press. Citado nas páginas 39 e 80.

Cussens, J. (1993). Bayes and pseudo-bayes estimates of conditional probabilities and their reliability. In *Eur. Conf. on Machine Learning (ECML 93)*, pag. 136–152. Citado na página 117.

Demšar, J. (2006). Statistical comparisonof classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30. Citado nas páginas 142 e 147.

Dietterich, T. G. (1990). Learning at knowledge level. In W.Shavlik, J. & Dietterich, T. G., editors, *Readings in machine learning*, pag. 11–25. Morgan Kauffmann. Citado na página 26.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pag. 1–15, Cagliari, IT. Springer. Citado na página 133.

Domingos, P. (1996). Unifying instance-based and rule-based induction. *Machine Learning*, 24(2):141–168. Citado na página 74.

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. *Knowledge Discovery and Data Mining*, 3(4):155–164. Citado nas páginas 89 e 111.

Domingos, P. (2002). Machine learning. In Klosgen, W. & Zytkow, J., editors, *Handbook of Data Mining and Knowledge Discovery*, pag. 660–670. Oxford University Press, New York. Citado na página 5.

Drummond, C. & Holte, R. C. (2003). C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. In *Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Data Sets (II)*. Citado na página 111.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern Classification*. Wiley, 2 edition. Citado na página 2.

Džeroski, S. & Lavrač, N. (2001). *Relational Data Mining*. Springer, Berlin Heidelberg (Germany). Citado na página 2.

Egan, J. P. (1975). *Signal Detection Theory and ROC Analysis*. Academic Press, New York, USA. Citado nas páginas 49 e 61.

Elkan, C. (2001a). The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'2001)*, pag. 973–978. Citado nas páginas 6, 53, e 90.

Elkan, C. (2001b). Learning and making decisions when costs and probabilities are both unknown. In *Inter. Conf. on Knowledge Discovery and Data Mining (KDD'01)*, pag. 204–213. Citado na página 117.

Everitt, B. S., Landau, S., & Leese, M. (2001). *Cluster Analysis*. Hodder Arnold, 4 edition. Citado na página 2.

Fawcett, T. (2003). ROC graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, HP Labs. `http://www.hpl.hp.com/personal/Tom_Fawcett/papers/HPL-2003-4.pdf`. Citado nas páginas 56 e 61.

Fawcett, T. & Flach, P. A. (2005). A response to webb and ting's On the application of ROC analysis to predict classification performance under varying class distributions. *Machine Learning*, 58(1):33–38. Citado na página 60.

Fawcett, T. & Provost, F. J. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316. Citado na página 88.

Flach, P. & Lavrač, N. (2003). Rule induction. In Berthold, M. & Hand, D., editors, *Intelligent Data Analysis*. Springer-verlag. Citado nas páginas 30, 32, e 37.

Flach, P. & Wu, S. (2005). Repairing concavities in ROC curves. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'2005)*, pag. 702–707, Edimburgo (UK). Citado na página 50.

Fleiss, J. L. (1981). *Statistical Methods for Rates and Proportions*. John Wiley & Sons, New York (USA), 2 edition. Citado na página 53.

Freitas, A. A. (1999). On rule interestingness measures. *Knowledge-Based Systems*, 12(5–6):309–315. Citado nas páginas 44 e 64.

Freund, Y., Iyer, R. D., Schapire, R. E., & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *J. of Machine Learning Res. (JMLR)*, 4:933–969. Citado nas páginas 137 e 152.

Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139. Citado na página 134.

Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54. Citado na página 73.

Fürnkranz, J. & Flach, P. (2005). ROC'n'rule learning – toward a better understanding of rule covering algorithms. *Machine Learning*, 58(1):39–77. Citado na página 75.

Fürnkranz, J. & Hüllermeier, E. (2003). Pairwise preference learning and ranking. In *14th European Conference on Machine Learning (ECML'2003)*, volume 2837 of *Lecture Notes in Computer Science*, pag. 145–156, Cavtat-Dubrovnik, Croatia. Springer. Citado na página 135.

Füurnkranz, J. & Widmer, G. (1994). Incremental reduced error pruning. In *Proceedings of the 11th Int. Conf. on Machine Learning*, pag. 70–77. Morgan Kaufmann. Citado na página 80.

Gastwirth, J. L. (1971). A general definition of the Lorenz curve. *Econometrica*, 39(6):1037–39. Citado na página 50.

Green, D. & Swets, J. A. (1989). *Signal Detection Theory and Psychophysics*. Peninsula Publishing, Los Altos, USA. Citado na página 49.

Han, H., Wang, W.-Y., & Mao, B.-H. (2005). Borderline-smote: A new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing*, volume 3644 of *Lecture Notes in Computer Science*, pag. 878–887. Springer. Citado na página 88.

Hand, D. J. & Till, R. (2001). A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45(2):171–186. Citado nas páginas 61 e 62.

Hanley, J. A. & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143:29–36. Citado na página 61.

Hart, P. E. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, IT-14:515–516. Citado na página 94.

Herbrich, R., Graepel, T., & Obermayer, K. (1999). Regression models for ordinal data: A machine learning approach. Technical Report 99/03, TU Berlin. Citado na página 135.

Hochberg, Y. & Tamhane, A. C. (1987). *Multiple Comparison Procedures*. Wiley. Citado nas páginas 81 e 118.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Arbor, MI, USA. Citado na página 21.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *National Academy of Sciences of the USA*, 79:2254–2258. Citado na página 20.

Hsu, J. (1996). *Multiple Comparisons: Theory and Methods*. Chapman. Citado nas páginas 81, 111, e 118.

Huang, J. & Ling, C. X. (2005). Dynamic ensemble re-construction for better ranking. In *Proceedings of 9th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'2005)*, volume 3721 of *Lecuture Notes in Artificial Intelligence*, Porto (PT). Springer. Citado na página 137.

Hussain, F., Liu, H., Suzuki, E., & Lu, H. (2000). Exception rule mining with a relative interesting measure. In *PAKDD-2000*, volume 1805 of *Lecture Notes in Artificial Intelligence*, pag. 86–97, Kyoto, Japan. Springer. Citado nas páginas 65 e 66.

Japkowicz, N., editor (2001). *AAAI Workshop on Learning from Imbalanced Data Sets*, Menlo Park, CA. AAAI Press. Techical report WS-00-05. Citado na página 88.

Japkowicz, N. (2003). Class imbalances: Are we focusing on the right issue? In Chawla et al. (2003). Citado nas páginas 98, 107, e 121.

Japkowicz, N. & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analasys*, 6(5):429–449. Citado na página 114.

Joachims, T. (1999). Making large-scale SVM learning practical. In Schölkopf, B., Burges, C., & Smola, A., editors, *Advances in Kernel Methods*. MIT-Press. Citado na página 141.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pag. 133–142. ACM Press. Citado na página 141.

Kaufman, K. A. & Michalski, R. S. (2005). From data mining to knowledge mining. In *Handbook in Statistics*, volume 24: Data Mining and Data Visualization, pag. 47–75. Elsevier. Citado nas páginas 3 e 5.

Kivinen, J., Mannila, H., & Ukknonen, E. (1994). Learning rules with local exceptions. In *Computational Learning Theory: EuroCOLT '93*, pag. 35–46, Oxford. Clarendon Press. Citado nas páginas 64 e 66.

Kowalsky, R. (1979). *Logic for Problem Solving*. North-Holland. Citado na página 29.

Kubat, M., Holte, R., & Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30(2-3):195–215. Citado na página 88.

Kubat, M. & Matwin, S. (1997). Addressing the course of imbalanced training sets: One-sided selection. In *Proceedings of 14th International Conference in Machine Learning*, pag. 179–186, San Francisco, CA. Morgan Kaufmann. Citado nas páginas 88 e 94.

Kuncheva, L. I. (2004). *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, Hoboken, New Jersey, USA. Citado na página 144.

Langley, P. (2000). Crafting papers on machine learning. In *Proceedings 17th International Conf. on Machine Learning*, pag. 1207–1212. Morgan Kaufmann, San Francisco, CA. Citado na página 3.

Laurikkala, J. (2001). Improving identification of difficult small classes by balancing class distributions. Technical Report A-2001-2, University of Tampere, Finland. Citado nas páginas 90, 95, e 97.

Lavrač, N. & Džeroski, S. (1994). *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, NY, USA. Citado na página 29.

Lavrač, N., Flach, P. A., & Zupan, B. (1999). Rule evaluation measures: A unifying view. In Džeroski, S. & Flach, P. A., editors, *9th International Workshop on Inductive Logic Programming (ILP-99)*, volume 1634 of *Lecture Notes in Computer Science*, pag. 174–185. Springer. Citado nas páginas 44 e 84.

Lavrač, N., Kavšek, B., Flach, P. A., & Todorovski, L. (2004). Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188. Citado na página 46.

Lee, H. D. (2005). Seleção de atributos importantes para a extração de conhecimento de bases de dados. Tese de Doutorado, ICMC-USP `http://www.teses.usp.br/teses/disponiveis/55/55134/tde-15032002-113112/`. Citado na página 5.

Lee, H. D., Monard, M. C., Voltolini, R. F., Prati, R. C., & Chung, W. F. (2006). A simple evaluation model for feature subset selection algorithms. In *Eighth Argentine Symposium on Artificial Intelligence (ASAI 2006)*. aceito. Citado nas páginas 9 e 154.

Leung, K. T. & Jr., D. S. P. (2003). Empirical comparisons of various voting methods in bagging. In *Proceedings 9th ACM SIGKDD Inter. Conf. on Knowledge Discovery and Data Mining (KDD'2003)*, pag. 595–600. Citado na página 138.

Ling, C. X., Huang, J., & Zhang, H. (2003). AUC: a statistically consistent and more discriminating measure than accuracy. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'2003)*, pag. 519–526. Morgan Kaufmann. Citado na página 61.

Ling, C. X. & Li, C. (1998). Data mining for direct mining: Problems and solutions. In *Proceedings of The Forth International Conference on Knowledge Discovery and Data Mining*, pag. 73–79. Citado nas páginas 88 e 111.

Liu, H., Lu, H., Feng, L., & Hussain, F. (1999). Efficient search of reliable exceptions. In *PAKDD'99*, volume 1574 of *Lecture Notes in Artificial Intelligence*, pag. 194–203. Springer. Citado na página 66.

Liu, H. & Motoda, H. (1998). *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, Massachusetts. Citado na página 5.

Manku, G. S. & Motwani, R. (2002). Approximate frequency counts over data streams. In *VLDB*, pag. 346–357. Citado na página 153.

Marzban, C. (2004). The ROC curve and the area under it as a performance measure. *Weather and Forecasting*, 19(6):1106–1114. Citado na página 97.

Matsubara, E. T., Monard, M. C., & Batista, G. E. A. P. A. (2005). Multi-view semi-supervised learning: An approach to obtain different views from text datasets. In *Advances in Logic Based Intelligent Systems*, volume 132, pag. 97–104. IOS Press. Citado na página 128.

Matsubara, E. T., Monard, M. C., & Prati, R. C. (2006). On the class-distribution labelling-step sensitivity of co-training. In *IFIP World Computer Congress - TC12 Artificial Intelligence in Theory and Practice (IFIP AI 2006)*. in print. Citado nas páginas 8, 127, e 151.

McCulloch, W. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–137. Citado na página 20.

Melo, V., Secato, M., & Lopes, A. A. (2003). Extração e identificação automática de informações bibliográficas de artigos científicos. In *IV Workshop on Advances and Trend in AI*, pag. 1–10, Chile. Citado na página 127.

Michalski, R. S. (1969). On the quasi-minimal solution of the covering problem. In *5th International Symposium on Information Processing (FCIP-69)*, volume A3, pag. 125–128, Bled, Yugoslavia. Citado nas páginas 18 e 48.

Michalski, R. S. (1983). A theory and methodology of inductive learning. In Michalski, R. S., Carbonell, J. G., & Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, pag. 111–161, Los Altos, CA. Morgan Kaufmann. Citado na página 25.

Michalski, R. S. (2004). Attributional calculus: A logic and representation language for natural induction. Technical Report MLI 04-2, George Mason University. `http://www.mli.gmu.edu/papers/2003-2004/mli04-2.pdf`. Citado na página 28.

Michie, D. (1988). Machine learning in the next five years. In Sleeman, D. H., editor, *Proceedings of the Third European Working Session on Learning*, pag. 107–122, Turing Institute, Glasgow. Pitman Publishing. Citado na página 25.

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). *Machine Learning, Neural and Statistical Classification*. Prentice Hall. Citado na página 2.

Minsky, M. S. & Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambrige, Mass. Citado na página 20.

Mitchell, T. (1997a). *Machine Learning*. McGraw Hill, New York, NY, USA. Citado na página 23.

Mitchell, T. M. (1997b). Does machine learning really work? *AI Magazine*, 18(3):11–20. Citado na página 36.

Monard, M. C. & Prati, R. C. (2005). Aprendizado de máquina simbólico para mineração de dados. In Castineira, M. I. & Schuhmacher, V. R. N., editors, *XIII Escola Regional de Informática (ERI 2005)*, pag. 1–26. WRCópias. Citado na página 9.

Mylne, K. R. (2002). Decision-making from probability forecasts based on forecast value. *Meteorological Applications*, 9:307–315. Citado na página 50.

Narayanan, A., Wu, X., & Yang, Z. R. (2002). Mining viral protease data to extract cleavage knowledge. *Bioinformatics*, 18(Suppl. 1):S5–S13. Citado nas páginas 68, 69, e 72.

Newman, D., Hettich, S., Blake, C., & Merz, C. (1998). UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`, University of California, Irvine, Dept. of Information and Computer Sciences. Citado nas páginas 3, 79, 105, 122, 127, e 140.

Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., & Brunk, C. (1994). Reducing misclassification costs. In *Proceedings of 11th International Conference in Machine Learning (ICML'1994)*, pag. 217–225. Citado na página 88.

Pednault, E. P. D., Rosen, B. K., & Apte, C. (2000). Handling imbalanced data sets in insurance risk modeling. Technical Report RC-21731, IBM Research Report. Citado na página 88.

Prati, R. C. (2003). O *Framework* de Integração do Sistema DISCOVER. Dissertação de Mestrado, ICMC-USP, `http://www.teses.usp.br/teses/disponiveis/55/55134/tde-20082003-152116/`. Citado na página 93.

Prati, R. C., Baranauskas, J. A., & Monard, M. C. (2001). Uma proposta de unificação da linguagem de representação de conceitos de algoritmos de aprendizado de máquina simbólicos. Technical Report 137, ICMC-USP. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_137.ps.zip`. Citado na página 32.

Prati, R. C., Baranauskas, J. A., & Monard, M. C. (2002). Padronização da sintaxe e informações sobre regras induzidas a partir de algoritmos de aprendizado de máquina simbólico. *Revista Eletrônica de Iniciação Científica*, 2(3). `http://www.sbc.org.br/reic/edicoes/2002e3`. Citado nas páginas 44 e 64.

Prati, R. C., Batista, G. E. A. P. A., & Monard, M. C. (2003a). Uma experiência no balanceamento artificial de conjuntos de dados para aprendizado com classes desbalanceadas utilizando análise ROC. In *IV Workshop de Inteligência Artificial (ATAI'2003)*. publicado em CD-ROM. Citado nas páginas 8 e 151.

Prati, R. C., Batista, G. E. A. P. A., & Monard, M. C. (2004a). Class imbalance versus class overlaping: An analysis of a learning system behaviour. In *Mexican International Conference on Artificial Intelligence (MICAI'2004)*, volume 2972 of *Lecture Notes in Artiricial Intelligence*, pag. 312–321, Mexico City (Mexico). Springer. Citado nas páginas 8, 98, e 151.

Prati, R. C., Batista, G. E. A. P. A., & Monard, M. C. (2004b). Learning with class skews and small disjuncts. In *Brazilian Symposium on Artificial Intelligence (SBIA'2004)*, volume 3171 of *Lecture Notes in Artificial Intelligence*, pag. 296–306, São Luiz (BR). Springer. Citado nas páginas 8, 121, e 151.

Prati, R. C., Batista, G. E. A. P. A., & Monard, M. C. (2006). Curvas ROC para a avaliação de classificadores. *IEEE América Latina*. submetido. Citado na página 9.

Prati, R. C. & Flach, P. A. (2004). ROCCER: A ROC hull rule learning algorithm. In Fürnkranz, J., editor, *ECML/PKDD 04 Workshop on Advances in Inductive Rule Learning*, pag. 144–153, Pisa (IT). Citado nas páginas 8, 74, e 150.

Prati, R. C. & Flach, P. A. (2005). ROCCER: An algorithm for rule learning based on ROC analysis. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'2005)*, pag. 823–828, Edinburgh, Scotland, UK. Citado nas páginas 8, 50, 74, e 150.

Prati, R. C., Geromini, M. R., & Monard, M. C. (2003b). An integrated environment for data mining. In *IV CONGRESS OF LOGIC APPLIED TO TECHNOLOGY (LAPTEC'2003)*, volume 2, pag. 55–62, Marília (BR). Pléiade. Citado na página 93.

Prati, R. C., Monard, M. C., & Carvalho, A. C. P. L. F. (2003c). Looking for exceptions on knowledge rules induced from HIV cleavage dataset. In *International Conference on Bioinformatics and Computational Biology - (ICoBiCoBi 2003)*. Resumo. Disponível em `http://www.vision.ime.usp.br/~cesar/programa/pdf/75.pdf`. Citado nas páginas 8 e 150.

Prati, R. C., Monard, M. C., & Carvalho, A. C. P. L. F. (2003d). A method for refining knowledge rules using exceptions. In *V Argentine Symposium on Artificial Intelligence (ASAI'2003)*, pag. 11, Buenos Aires, AR. Citado nas páginas 8 e 150.

Prati, R. C., Monard, M. C., & Carvalho, A. C. P. L. F. (2003e). Refinando regras de conhecimento por meio de exceções. In *IV Encontro Nacional de Inteligencia Artificial (ENIA'2003)*, volume VII, pag. 477–486, Campinas (BR). Sociedade Brasileira de Computação (SBC). Prêmio de melhor trabalho do evento. Citado nas páginas 8 e 150.

Prati, R. C., Monard, M. C., & Carvalho, A. C. P. L. F. (2004c). A method for refining knowledge rules using exceptions. *SADIO Electronic Journal of Informatics and Operations Research*, 6(1):53–65. Seleção dos melhores trabalhos do ASAI'2003. Citado nas páginas 8 e 150.

Prati, R. C., Monard, M. C., & Carvalho, A. C. P. L. F. (2004d). Refinando regras de conhecimento por meio de exceções. *Scientia*, 14(2):115–132. Seleção dos melhores trabalhos do ENIA'2003. Citado nas páginas 8 e 150.

Prati, R. C., Monard, M. C., & Carvallho, A. C. P. L. F. (2004e). Looking for exceptions on knowledge rules induced from HIV cleavage data set. *Genetics and Molecular Biology*, 27(4):637–643. Versão expandida do resumo publicado no ICoBiCoBi'2003. Citado nas páginas 8 e 150.

Provost, F. & Fawcett, T. (2001). Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231. Citado nas páginas 57 e 75.

Provost, F., Fawcett, T., & Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. In *Proceedings Fifteenth International Conference on Machine Learning (ICML'1998)*, pag. 445–453, Madison, WI. Morgan Kaufmann, San Francisco, CA. Citado na página 57.

Provost, F. J. & Domingos, P. (2003). Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215. Citado nas páginas 83, 97, e 109.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106. Citado nas páginas 18 e 73.

Quinlan, J. R. (1987). Generating production rules from decision trees. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI'1987)*, pag. 304–307, Italy. Citado nas páginas 18 e 19.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann. Citado nas páginas 19, 42, 43, 80, 97, e 141.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan, New York. Citado na página 20.

Rudin, C., Cortes, C., Mohri, M., & Schapire, R. E. (2005). Margin-based ranking meets boosting in the middle. In *18th Conference on Learning Theory*, volume 3559 of *Lecture Notes in Computer Science*, pag. 63–78, Bertinoro, Italy. Springer. Citado na página 137.

Rumelhart, D. E. & McClelland, J. L. (1986). *Parallel Distributed Processing*. MIT Press, Cambridge, MA. Citado na página 20.

Russell, S. & Norvig, P. (2003). *Artificial Intelligence – A Modern Approach*. Prentice Hall, 2 edition. Citado na página 17.

Ryabko, D. (2006). Pattern recognition for conditionally independent data. *Journal of Machine Learning Research*, 7(Apr):645–664. Citado na página 6.

Schapire, R. E. (2001). *The Design and Analysis of Efficient Learning Algorithms*. The MIT Press, Cambridge, Massachusetts. Citado na página 23.

Spackman, K. A. (1989). Signal detection theory: Valuable tools for evaluating inductive learning. In *Proceedings of the 6th Int Workshop on Machine Learning (ICML'1989)*, pag. 160–163. Morgan Kaufmann. Citado na página 50.

Stanfill, C. & Waltz, D. L. (1986). Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228. Citado na página 20.

Stolfo, S. J., Fan, D. W., Lee, W., Prodromidis, A. L., & Chan, P. K. (1997). Credit card fraud detection using meta-learning: Issues and initial results. In *AAAI-97 Workshop on AI Methods in Fraud and Risk Management*. Citado na página 88.

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press. Citado na página 2.

Suzuki, E. (1997). Autonomous discovery of reliable exception rules. In *KDD´97)*, pag. 159–176. AAAI Press. Citado nas páginas 64 e 66.

Tang, L. & Liu, H. (2005). Bias analysis in text classification for highly skewed data. In *IEEE International Conference on Data Mining (ICDM 2005)*, pag. 781–784, Houston, Texas, USA. IEEE Computer Society. Citado na página 88.

Utgoff, P. E. (1989). Incremental induction of decision trees. *Machine Learning*, 4(2):161–186. Citado na página 22.

van den Eijkel, G. (2003). Information-theoretic tree and rule induction. In *Intelligent Data Analysis*, pag. 465–475. Springer. Citado na página 74.

van Erp, M., Vuurpijl, L., & Schomaker, L. (2002). An overview and comparison of voting methods for pattern recognition. In *Proceedings of the 8th Int Workshop on Frontiers in Handwriting Recognition*, pag. 195–200, Niagara-on-the-Lake, Canada. Citado nas páginas 137 e 138.

van Rijsbergen, C. J. (1979). *Information Retrieval*. University of Glasgow. `http://www.dcs.gla.ac.uk/Keith/Preface.html`. Citado na página 46.

Webb, G. I. & Ting, K. M. (2005). On the application of ROC analysis to predict classification performance under varying class distributions. *Machine Learning*, 58(1):25–32. Citado na página 59.

Weiss, G. M. (1995). Learning with rare cases and small disjucts. In *International Conference on Machine Learnin (ICML'1995)*, pag. 558–565. Citado na página 122.

Weiss, G. M. (2003). *The Effect of Small Disjuncts and Class Distribution on Decision Tree Learning*. PhD thesis, Rutgers University. Citado nas páginas 121, 122, e 124.

Weiss, G. M. (2004). Mining with rarity: a unifying framework. *SIGKDD Explorations*, 6(1):7–19. Citado na página 88.

Weiss, G. M. & Provost, F. J. (2003). Learning when training data are costly: The effect of class distribution on tree induction. *J. Artif. Intell. Res. (JAIR)*, 19:315–354. Citado nas páginas 88, 100, e 110.

Widmer, G. & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101. Citado na página 5.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Communications*, 2(3):408–421. Citado na página 95.

Witten, I. & Frank, E. (2000). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufman. Citado na página 30.

Zadrozny, B. & Elkan, C. (2001). Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'2001)*, pag. 204–213. ACM Press. Citado na página 109.

Zhou, X.-H., McClish, D. K., & Obuchowski, N. A. (2002). *Statistical Methods in Diagnostic Medicine*. John Wiley & Sons Inc. Citado na página 49.