Aprendizado de máquina semi-supervisionado: proposta de um algoritmo para rotular exemplos a partir de poucos exemplos rotulados

Marcelo Kaminski Sanches

Aprendizado de máquina semi-supervisionado: proposta de um algoritmo para rotular exemplos a partir de poucos exemplos rotulados[1]

Marcelo Kaminski Sanches

Orientadora: *Profª Drª Maria Carolina Monard*

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

USP – São Carlos
Junho/2003

[1]Trabalho realizado com auxílio financeiro da CAPES

Este texto foi preparado com o formatador de textos LaTeX. O sistema de citações de referências bibliográficas utiliza o padrão *Chicago* do sistema BibTeX.

Dedico esta dissertação aos meus pais, que sem medir esforços sempre estiveram determinados na minha educação. Apoio esse incondicional que me deu a possibilidade de me dedicar de forma integral e com total tranqüilidade aos estudos. Amo vocês!

Agradecimentos

A Deus, pela sua onipresença.

À minha família, meus pais Helena e Manoel e meus irmãos Daniele e Adriano, por todo amor e carinho recebidos.

À professora Maria Carolina, pela forma zelosa de orientar este mestrado e por me ensinar muito mais que Aprendizado de Máquina. Obrigado pela confiança em mim depositada e pelo cuidado com este trabalho. Obrigado também pelos conselhos, pelos nossos papos, enfim, obrigado por estar sempre disposta a ajudar.

Aos professores da UEPG, em especial ao professor Bráulio Coelho Ávila, por tanto tempo meu orientador de iniciação científica, e, por conseguinte, um dos responsáveis pela minha incursão na área científica.

Aos amigos Lilian, Luciano, Márcio e Omar, pela ajuda recebida quando cheguei em São Carlos.

Aos amigos Douglas (Indomada), Jacqueline (Jackie), Renato (Japa), Rogério (Rodgers), Vinícius (Cuhia) e o pessoal do LABIC, e todos os outros amigos que fiz em São Carlos, pelos bons momentos passados juntos, fundamentais para o bom andamento deste trabalho.

Aos funcionários e professores do ICMC, pela atenção e dedicação.

À USP pela oportunidade.

À Capes pelo apoio financeiro.

E a todos que de alguma maneira contribuíram para este trabalho.

Resumo

A fim de se utilizar algoritmos de Aprendizado de Máquina para tarefas de classificação, é admitida a existência de um conjunto de exemplos rotulados, conhecido como conjunto de treinamento, o qual é utilizado para o treinamento do classificador. Entretanto, em casos reais, esse conjunto de treinamento pode não conter um número de exemplos suficientemente grande para se induzir um bom classificador.

Recentemente, a comunidade científica tem mostrado um grande interesse em uma variação dessa abordagem de aprendizado supervisionado. Essa nova abordagem, conhecida como aprendizado semi-supervisionado, assume que, juntamente com o conjunto de treinamento, há um segundo conjunto, de exemplos não rotulados, também disponível durante o treinamento. Uma das metas do aprendizado semi-supervisionado é o treinamento de classificadores quando uma grande quantidade de exemplos não rotulados está disponível juntamente com um pequeno conjunto de exemplos rotulados.

A motivação para o aprendizado semi-supervisionado deve-se ao fato que, em muitas aplicações do mundo real, conjuntos de exemplos não rotulados são facilmente encontrados ou muito baratos para serem coletados, quando comparados aos conjuntos de exemplos rotulados. Um outro fator é que exemplos não rotulados podem ser coletados de forma automática enquanto os rotulados necessitam de especialistas ou outros custosos recursos de classificação.

Os exemplos não rotulados podem ser utilizados de diversas maneiras. Neste trabalho é explorado um mecanismo no qual os exemplos não rotulados podem ser utilizados para melhorar tarefas de classificação e é proposto um algoritmo semi-supervisionado, denominado $k\text{-}means_{ki}$, o qual viabiliza o uso de exemplos não rotulados em aprendizado supervisionado.

A técnica utilizada pelo algoritmo proposto está baseada em duas premissas. A primeira delas é que os exemplos tendem a se agrupar naturalmente em clusters, ao invés de se distribuirem uniformemente no espaço de descrição dos exemplos. Além disso, cada exemplo do conjunto inicial de exemplos rotulados deve estar localizado perto do centro de um dos clusters existentes no espaço de descrição de exemplos. A segunda premissa diz que a maioria dos exemplos nos clusters pertencem a uma classe

específica. Obviamente, a validade dessas premissas é dependente do conjunto de dados utilizado.

O algoritmo k-$means_{ki}$ funciona bem nos casos em que os dados estão em conformidade com ambas as premissas. Entretanto, caso elas sejam violadas, a performance do algoritmo não será boa. São mostrados experimentos utilizando conjuntos de dados do mundo real, escolhendo-se aleatoriamente exemplos desses conjuntos para atuarem como exemplos rotulados.

Abstract

In order to apply machine learning classification algorithms, it is assumed that there exists a set of labeled data, called the training set, which is used to train the classifier. However, in real life, this training set may not contain enough labeled data to induce a good classifier.

Recently, there has been much interest in a variant of this approach. This new approach, known as semi-supervised learning, assumes that, in addition to the labeled training data, there exists a second set of unlabeled data which is also available during training. One of the goals of semi-supervised learning is training classifiers when large amounts of unlabeled data are available together with a small amount of labeled data.

The appeal of semi-supervised learning is due to the fact that in many real-world applications, such sets of unlabeled data are either readily available or inexpensive to collect compared to labeled data. Unlabeled data can often be collected by automated means while labeled data requires human experts or other limited or expensive classification resources that may even be unfeasible in some cases.

There are several ways in which unlabeled data can be used. In this work we explore one mechanism by which unlabeled data can be used to improve classification problems and propose a semi-supervised algorithm, called $k\text{-}means_{ki}$, for using unlabeled data for supervised learning.

There are two premises underlying the technique used by the proposed algorithm. The first is that input data falls naturally into clusters rather than being uniformly distributed across the entire input space. Furthermore, the initial input labeled data should fall near the center of the existing clusters in the input space. The second premise is that many of the points in some of these clusters belong to specific output categories. Obviously, the validity of these premises is dependent on the dataset used.

The $k\text{-}means_{ki}$ algorithm works well when the data conform to both premises. If these assumptions are violated poor performance can result. Experiments using real world datasets where we randomly select a subset of the available data to act as labeled exemplars are shown.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo

1

Introdução

"Todo homem que se vende recebe mais do que vale."

Barão de Itararé

O desenvolvimento tecnológico tem proporcionado novas formas de gerar e coletar dados como, por exemplo, código de barras em produtos e automatização de negócios e transações. Além disso, avanços no armazenamento de dados, tais como velocidade, capacidade de acesso e baixo custo, também impulsionaram a geração de grandes quantidades de dados, tornando-se evidente a necessidade de se gerar técnicas e ferramentas que auxiliem de forma automática e inteligente, os seres humanos na tarefa de análise, e, por conseqüência, extração de informações úteis (conhecimento) desses dados.

Uma área que está sendo bastante utilizada na tarefa de extração de conhecimento em grandes volumes de dados é a área de Aprendizado de Máquina (AM) (Mitchell, 1997). Aprendizado de Máquina se preocupa com o desenvolvimento de modelos computacionais que adquirem conhecimento utilizando fatos e conhecimento de fundo[1]

[1] *Background knowledge*

disponíveis. Suas técnicas e seus algoritmos estão cada vez mais presentes em processos como Mineração de Dados[2] (MD) (Rezende, Pugliesi, Melanda, & Paula, 2003), que é uma das etapas de *Knowledge Discovery in Databases* (KDD) (Fayyad, Piatetsky-Shapiro, Smyth, & Uthurusamy, 1996).

Aprendizado de Máquina pode ser caracterizado como aprendizado supervisionado e não supervisionado. Caso os exemplos contidos nas bases de dados estejam rotulados com sua classe correspondente, pode-se utilizar algoritmos de aprendizado supervisionado, os quais induzem padrões a partir dos dados. Mas, no caso dos exemplos não estarem rotulados, faz-se necessária a utilização de algoritmos de aprendizado não supervisionado. Esses algoritmos buscam por padrões nos dados a partir de uma caracterização de similaridade. Tais padrões são chamados clusters, sendo que os exemplos contidos em um mesmo cluster são mais similares, segundo alguma medida de similaridade, do que os exemplos contidos em clusters diferentes. Assim, o processo de formação desses clusters, geralmente conhecido como *clustering*[3], é visto como um processo que agrupa exemplos de forma física ou abstrata, em clusters de exemplos similares (*clustering* clássico). Uma outra abordagem mais recente considera que os exemplos devem ser agrupados não apenas porque eles são semelhantes segundo uma certa medida de similaridade, mas também porque eles possuem um certo significado conceitual. Tal abordagem é conhecida como *clustering* conceitual.

1.1 Motivação

Como mencionado, aprendizado supervisionado trabalha com conjuntos de dados cujos exemplos estão pré-classificados, *i.e.*, estão rotulados com o atributo "classe". O objetivo do aprendizado supervisionado é construir um modelo preditivo, que consiste em rotular novos exemplos que não possuem o atributo classe associado. Assim, a partir dos exemplos rotulados, denominados exemplos de treinamento, um algoritmo de aprendizado é utilizado para induzir padrões presentes em cada classe, gerando uma descrição de cada classe. Essas descrições são utilizadas para predizer a classe de novos exemplos.

No caso de aprendizado não supervisionado utilizando o processo de *clustering*, o problema é agrupar os exemplos não rotulados que apresentem um mesmo padrão de acordo com algum critério pré-estabelecido. Na abordagem clássica de *clustering*, os clusters são determinados exclusivamente com base em uma medida de similaridade pré-

[2] *Data Mining*

[3] O termo em inglês *clustering* será utilizado neste trabalho por ser amplamente difundido na comunidade científica.

definida. Após determinar os atributos relevantes para descrever os exemplos, vetores de valores desses atributos são utilizados para descrever todo o conjunto de exemplos. Considerando os atributos como dimensões de um espaço multidimensional, a descrição de cada exemplo corresponde a um ponto no espaço. A similaridade entre eles pode ser medida de acordo com uma função que calcula a distância entre os pontos (exemplos) no espaço de descrição de exemplos. Como a distância é uma função que envolve somente os atributos relativos a dois exemplos, *clustering* baseado em similaridade pode ser feito de forma relativamente simples e sem necessitar de algum tipo de conhecimento sobre o assunto tratado. A abordagem baseada em similaridade tem produzido eficientes algoritmos de *clustering*, que têm sido úteis em várias aplicações (Guha, Rastogi, & Shim, 1998; Huang, 1997; Jain, Murty, & Flynn, 1999; McCallum, Nigam, & Ungar, 2000; Michalski & Stepp, 1992).

Entretanto, essa abordagem sofre de algumas limitações significantes. Diferentemente do que ocorre com o aprendizado supervisionado, os resultados de um processo de *clustering* não fornecem nenhum tipo de explicação ou descrição, mas apenas clusters. Porém, muitas vezes, não se está interessado apenas nos clusters, mas também em alguma explicação ou descrição conceitual dos exemplos que foram agrupados em um mesmo cluster.

Para superar isso é necessário que se faça uma interpretação dos clusters encontrados para tentar extrair o significado conceitual deles, o que não é uma tarefa fácil. De fato, isso pode ser mais difícil do que a própria geração dos clusters. Isso porque induzir as descrições das classes dos exemplos é uma tarefa inferencial complexa. Assim, clusters gerados exclusivamente com base em medidas numéricas de similaridade pré-definidas podem não ser apropriados para uma explicação conceitual dos exemplos que pertencem a um cluster.

Dessa forma, existe a preferência, sempre que exista a possibilidade, de se escolher aprendizado supervisionado. Entretanto, em muitos casos do mundo real, geralmente o número de exemplos rotulados é muito pequeno, quando não inexistente, e classificadores induzidos a partir de um pequeno conjunto de exemplos rotulados apresentam, geralmente, baixa precisão. Tais fatores inviabilizam o uso de algoritmos de aprendizado supervisionado nesses casos. Uma alternativa seria rotular manualmente cada exemplo. Porém, esse processo é extremamente caro e, às vezes, impossível, dado o imenso volume de dados. Assim, torna-se interessante a existência de mecanismos que possam rotular exemplos de forma automática, com o objetivo de viabilizar o uso de aprendizado supervisionado em situações nas quais o número de exemplos rotulados é pequeno e o número de exemplos não rotulados é grande. Essa é uma das metas do aprendizado semi-supervisionado, objeto de estudo deste trabalho.

1.2 Objetivos

Como mencionado, exemplos rotulados são escassos e de geração cara. Entretanto, pedir a um especialista que rotule um pequeno conjunto de exemplos é uma tarefa aceitável. Nesse caso, pode ser considerado o uso de aprendizado semi-supervisionado para incrementar o conjunto de exemplos rotulados a ser utilizado por algoritmos de aprendizado supervisionado. Neste trabalho é proposto um algoritmo, denominado *k-means*$_{ki}$, que pode ser utilizado no processo de rotulação de exemplos a serem utilizados em tarefas de classificação. O algoritmo proposto tem como requisito básico a existência de um pequeno conjunto de exemplos representativos[4] rotulados, por um especialista, com um alto grau de certeza.

Baseado no conhecido algoritmo *k-means* (Macqueen, 1967), o algoritmo *k-means*$_{ki}$ visa incrementar o conjunto de exemplos rotulados, com o intuito de melhorar o processo de classificação. A principal diferença entre os dois algoritmos se encontra na fase de seleção de centróides. Ao passo que o *k-means* seleciona aleatoriamente seus centróides iniciais, o *k-means*$_{ki}$ define, como centróides iniciais, cada um dos exemplos rotulados disponíveis.

Uma outra diferença é quanto ao processo de *clustering* propriamente dito. No algoritmo *k-means* cada exemplo é associado ao cluster (centróide) mais próximo, já no algoritmo *k-means*$_{ki}$ é previamente estipulado um *threshold* t, e cada exemplo somente poderá ser associado a um dado cluster caso esteja a uma distância menor ou igual a t de seu respectivo centróide.

Uma vez que cada cluster é originado por um exemplo rotulado, a idéia é rotular todos os exemplos associados a um dado cluster com a classe do exemplo que o originou, incrementando dessa forma o conjunto de exemplos rotulados.

Deve ser observado que este é o primeiro trabalho sobre aprendizado semi-supervisionado desenvolvido no grupo de pesquisa de Inteligência Computacional do LABIC [5], o qual será integrado a um ambiente computacional em desenvolvimento denominado DISCOVER, descrito brevemente a seguir.

1.2.1 Projeto DISCOVER

Eventuais problemas decorrentes da utilização de algoritmos de aprendizado reimplementados conduzem, muitas vezes, à necessidade de utilização dos algoritmos de aprendizado tais como foram implementados pelos seus idealizadores, e, por conseguinte,

[4] Um exemplo representativo é aquele que representa uma instância de um determinado conceito com bastante certeza.

[5] Laboratório de Inteligência Computacional — http://labic.icmc.usp.br

todas as atividades necessárias para a execução dos experimentos devem ser feitas para cada algoritmo. Essa necessidade implica no desenvolvimento de programas para automatizar essas tarefas.

Muitos pesquisadores em nosso laboratório de pesquisa — LABIC — utilizam esses algoritmos em suas pesquisas e, muitas vezes, reimplementam algum tipo de código semelhante para a realização de experimentos.

Esses fatores levaram alguns pesquisadores do laboratório a propor e implementar, uma série de ferramentas para facilitar a configuração e execução de experimentos. Surgiu então a proposta (Baranauskas & Batista, 2000) de criar um projeto no qual todos os membros do laboratório estariam envolvidos. A esse projeto foi dado o nome de DISCOVER.

Um dos principais objetivos do projeto DISCOVER é tentar diminuir o esforço, por parte dos membros do projeto, necessário para implementar um experimento. Muitas vezes, um programa semelhante é implementado diversas vezes, por membros diferentes, por falta de comunicação entre os membros ou por falta de documentação dos programas já implementados. Ainda, é comum que diversas implementações sejam perdidas quando seus autores se desligam do laboratório, após finalizar o curso.

A princípio, o projeto DISCOVER consistiria apenas de um repositório de *scripts*. Posteriormente surgiu a proposta de criar um ambiente integrado, ao qual foi dado o nome de ambiente DISCOVER, em que os *scripts* seriam substituídos por bibliotecas de classes e essas bibliotecas empacotadas como componentes, com a composição dos componentes sendo feita por meio de uma interface gráfica (Geromini, 2002).

Para se tornar um ambiente integrado, várias questões que dizem respeito ao gerenciamento do processo, comunicação entre os participantes e interação entre os diferentes componentes devem ser respondidas. Para ajudar a responder essas questões foi feito um estudo para fazer do processo de implementação do ambiente DISCOVER um processo de Engenharia de Software (Rozante, 2003).

Além disso, um outro ponto importante diz respeito à arquitetura do ambiente. Prati (2003) propõe um *framework* para a integração dos componentes do ambiente DISCOVER baseado em padrões de software, no qual os componentes são integrados por meio de uma linguagem baseada em XML, a qual foi dada o nome de xDML.

Inicialmente, o ambiente DISCOVER tem como principal objetivo fornecer um campo de prova para os pesquisadores do laboratório. A vantagem do ambiente DISCOVER como ferramenta de apoio à pesquisa em KDD, em relação a outros sistemas, é a visão unificada que os formatos padrões oferecem para quem está desenvolvendo novos componentes, além de um conjunto de ferramentas de manipulação dos mesmos.

Atualmente, existem definidas sintaxes padrões para a representação de dados e para

a representação de conhecimento de diversos indutores simbólicos (Prati, Baranauskas, & Monard, 2001), bem como bibliotecas que oferecem funcionalidades sobre essas sintaxes padrão. Novas sintaxes padrão estão sendo especificadas, principalmente para a representação de regras de regressão (Dosualdo, 2003) e regras de associação (Melanda, 2002).

No caso de pré-processamento de dados, atividade fundamental no processo de KDD, na qual é gasto a maior parte do tempo e do esforço despendido em todo o processo de KDD, foi projetado e implementado o ambiente DISCOVER LEARNING ENVIRONMENT — DLE. O ambiente DLE é composto por dois módulos, a biblioteca de classes DISCOVER OBJECT LIBRARY — DOL (Batista & Monard, 2003b) e o ambiente para gerenciamento de experimentos SNIFFER (Batista & Monard, 2003a).

A biblioteca DOL provê um conjunto de métodos de pré-processamento de dados. O ambiente de gerenciamento de experimentos SNIFFER é um ambiente capaz de automatizar a execução de experimentos, facilitando a comparação de diferentes métodos e a publicação de resultados. O presente trabalho fez uso da DOL — para pré-processar os dados utilizados pelo algoritmo $k\text{-}means_{ki}$ — e do SNIFFER — na indução, e comparação, dos classificadores induzidos durante a etapa experimental.

Algumas outras pesquisas realizadas e em curso em nosso laboratório também resultaram em diversos trabalhos e idéias sobre novas ferramentas para o projeto DISCOVER, relacionadas a Aprendizado de Máquina (Bernardini, 2002; Caulkins, 2000; Gomes, 2002; Milaré, 2003; Pila, 2001), Mineração de Dados (Baranauskas, 2001; Batista, 1997, 2003; Félix, 1998; Horst, 1999; Lee, 2000; Nagai, 2000; Paula, 2003; Pugliesi, 2001) e Mineração de Textos[6](MT) (Imamura, 2001; Martins, 2003).

1.3 Organização da Dissertação

Este trabalho está dividido em sete capítulos, sendo esta Introdução o primeiro deles. Os demais capítulos estão organizados da seguinte forma:

- no Capítulo 2 é apresentada uma visão geral de Aprendizado de Máquina, bem como alguns conceitos e definições do aprendizado supervisionado;

- no Capítulo 3 é abordado o processo de *clustering* — aprendizado não supervisionado, algumas técnicas e algoritmos são apresentados, bem como algumas medidas de similaridade utilizadas nesse processo;

[6] *Text Mining*

- no Capítulo 4 é dada uma visão geral sobre aprendizado semi-supervisionado, área relativamente recente de pesquisa em AM que propõe o uso de exemplos rotulados e não rotulados no processo de aprendizado;
- no Capítulo 5 é apresentado o algoritmo $k\text{-}means_{ki}$, fruto do presente trabalho, a ser utilizado no processo de rotulação de exemplos a partir de um pequeno conjunto de exemplos rotulados;
- no Capítulo 6 são apresentados os resultados experimentais obtidos com o algoritmo $k\text{-}means_{ki}$;
- no Capítulo 7 são apresentadas as conclusões do trabalho, bem como algumas sugestões de trabalhos futuros.

Capítulo

2

Aprendizado de Máquina

" Se quiser por à prova o caráter de um homem, dê lhe poder."

Abrahan Lincoln

Aprendizado de Máquina é uma sub-área da Inteligência Artificial (IA), cujo objetivo é desenvolver métodos, técnicas e ferramentas para construir máquinas inteligentes, que se modificam para realizar cada vez melhor sua(s) tarefa(s). "Melhor" nesse sentido pode ser trabalhar com mais eficiência ou precisão, ou ainda de forma a abranger uma classe maior de problemas. Para tanto, essas máquinas devem ser capazes de aprender (Mitchell, 1997).

Neste capítulo é apresentada uma visão geral de Aprendizado de Máquina.

2.1 Introdução

A capacidade de aprender é essencial para um comportamento inteligente, assim, AM torna-se uma área de pesquisa muito importante em IA. Aprendizado de Máquina

engloba estudos de métodos computacionais para adquirir novos conhecimentos, novas habilidades e novos meios de organizar o conhecimento já existente. O estudo de técnicas de aprendizado baseado em computador também pode fornecer um melhor entendimento do próprio processo de raciocínio humano.

Um sistema capaz de aprender é aquele que se modifica automaticamente no sentido de que ele possa fazer a(s) mesma(s) tarefa(s) sobre um mesmo domínio de conhecimento, de uma maneira cada vez mais eficiente (Simon, 1983). Para aprender, o sistema, bem como os seres humanos, podem se valer de estratégias de aprendizado. Algumas dessas estratégias são sucintamente descritas a seguir.

Aprendizado por hábito: Nesse tipo de aprendizado o aprendiz não precisa realizar nenhuma inferência sobre a informação fornecida, sendo o conhecimento diretamente assimilado pelo aprendiz. Essa estratégia inclui o aprendizado por memorização direta de descrições de um dado conceito.

Aprendizado por instrução: No aprendizado por instrução o aprendiz adquire conceitos de uma fonte, como por exemplo um professor, uma publicação ou um livro, mas não copia diretamente a informação fornecida para a memória, como no caso anterior. O aprendizado engloba a seleção dos fatos mais relevantes e/ou a transformação da informação fonte em formas mais apropriadas.

Aprendizado por dedução: A dedução é definida como inferência logicamente correta, *i.e.*, a conclusão obtida de várias premissas iniciais verdadeiras sempre preserva a verdade. A inferência atua somente nas informações contidas no conhecimento, nada além disso (Raggett & Bains, 1992).

O aprendiz adquire um conceito por meio de dedução sobre o conceito já adquirido. Ou seja, essa estratégia inclui um processo no qual o conhecimento aprendido é o resultado de uma transformação sobre um conhecimento já possuído, que preserva veracidade. Em geral, no aprendizado dedutivo realiza-se uma seqüência de deduções ou cálculos sobre a informação presente, memorizando, em alguns casos, o resultado.

Aprendizado por analogia: Nessa estratégia, o aprendiz adquire um novo conceito modificando a definição de um conceito semelhante já conhecido. Em vez de formular ao acaso uma regra para um novo conceito, adapta-se uma regra existente modificando-a apropriadamente de forma a poder aplicá-la ao novo conceito.

Por exemplo, se alguém tem o conceito de uma laranja, o aprendizado do conceito de uma tangerina pode ser muito simples, apenas notando as semelhanças e as diferenças entre as duas.

O aprendizado por analogia pode ser visto como a combinação do aprendizado indutivo, apresentado na próxima seção, com o dedutivo. Por meio de inferência indutiva, determinam-se características gerais unindo os conceitos que estão sendo comparados. Depois, por inferência dedutiva e a partir dessas características, determina-se o conceito a ser aprendido.

Uma outra estratégia de aprendizado, denominada aprendizado por indução, ou aprendizado indutivo, é provavelmente a mais utilizada para aprender novos conceitos. Essa estratégia, utilizada neste trabalho, é descrita com maiores detalhes na próxima seção.

2.2 Aprendizado Indutivo

A generalização de conceitos a partir de alguns fatos, e a descoberta de padrões em coleções de observações aparentemente caóticas, é atingida pelo aprendizado por indução, ou aprendizado indutivo.

Indução é a forma de inferência lógica que permite que conclusões genéricas sejam obtidas a partir de conjuntos de fatos ou observações (exemplos) particulares. Ela é caracterizada como o raciocínio que parte do específico para o geral, do particular para o universal, da parte para o todo. Ou seja, o conhecimento generalizado extrapola aquele contido nos fatos e ele pode ou não preservar a verdade.

Apesar disso, a inferência indutiva é um dos principais métodos utilizados para derivar conhecimento novo e predizer eventos futuros, sendo ela a responsável por inúmeras descobertas da humanidade. Ao utilizá-la, deve-se ter muita cautela, pois se o número de exemplos for insuficiente, ou se os exemplos não forem muito bem escolhidos, as generalizações realizadas para induzir hipóteses podem ser de pouco valor. O algoritmo de aprendizado, responsável pela geração de hipóteses, é chamado *indutor*.

O aprendizado indutivo é efetuado a partir de raciocínio sobre exemplos fornecidos por um processo externo ao sistema de aprendizado, e pode ser dividido em:

1. *aprendizado supervisionado* e
2. *aprendizado não supervisionado.*

No aprendizado supervisionado, cada exemplo é descrito pelos valores de um conjunto de atributos e possui, obrigatoriamente, uma classe associada. A classe, também conhecida como rótulo do exemplo, é um atributo especial que descreve uma instância do fenômeno de interesse, *i.e.*, o conceito que se deseja induzir. A idéia geral consiste

em induzir um conceito utilizando esses exemplos rotulados, denominado conjunto de exemplos de treinamento, realizando generalizações e especializações, tal que o conceito (hipótese) induzido é capaz de predizer a classe (rótulo) de futuros exemplos. Nesse caso, o conceito induzido é visto como um *classificador*.

Uma outra observação relacionada ao aprendizado supervisionado diz respeito ao tipo do atributo classe. Caso ele seja contínuo, o problema de indução é conhecido como *regressão* e caso a classe seja discreta, o problema é conhecido como *classificação*. O aprendizado supervisionado será abordado em maiores detalhes na Seção 2.6 na página 19.

Já no aprendizado não supervisionado, os exemplos não possuem uma classe correspondente. Nesse caso o indutor analisa os exemplos fornecidos e tenta determinar se alguns deles podem ser agrupados de alguma maneira, formando agrupamentos ou clusters (Cheeseman & Stutz, 1990). Após a determinação dos agrupamentos, normalmente, é necessária uma análise para determinar o que cada agrupamento significa no contexto do problema que está sendo analisado. Esse tipo de aprendizado será melhor detalhado na Seção 2.7 na página 24 e no Capítulo 3 na página 27.

Portanto, a escolha de qual tipo de aprendizado indutivo (supervisionado ou não supervisionado) utilizar, depende dos exemplos estarem ou não rotulados com o atributo classe. Na Figura 2.1 é mostrada a hierarquia do aprendizado indutivo.

Figura 2.1: Hierarquia do aprendizado indutivo

2.3 Paradigmas de Aprendizado

Uma vez definida a estratégia de aprendizado a ser utilizada, o próximo passo pode ser a escolha do paradigma a ser empregado. Há diversos paradigmas de aprendizado

propostos na literatura. Alguns desses paradigmas serão brevemente descritos a seguir.

Paradigma simbólico: os sistemas de aprendizado simbólico buscam aprender construindo representações simbólicas de um conceito pela análise de exemplos e contra-exemplos desse conceito. As representações simbólicas estão tipicamente na forma de alguma expressão lógica, árvore de decisão, regra de produção ou rede semântica, e serão detalhadas na Seção 2.5 na página 15.

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, conhecido como particionamento recursivo, foram estudados por pesquisadores da área de IA e Estatística.

Os trabalhos com indução de regras de decisão surgiram com a simples tradução das árvores de decisão para regras, com a poda realizada sobre elas. Tal abordagem foi inicialmente proposta por Quinlan (1987). Posteriormente, foram criados métodos para induzir regras diretamente a partir dos dados (Michalski, Mozetic, Hong, & Lavrac, 1986).

Paradigma estatístico: pesquisadores em estatística têm criado diversos métodos de classificação, muitos deles semelhantes aos métodos empregados em AM. Como regra geral, técnicas estatísticas tendem a focar tarefas em que todos os atributos têm valores contínuos ou ordinais. Muitos deles também são paramétricos, assumindo alguma forma de modelo, e então encontrando valores apropriados para os parâmetros do modelo a partir de dados. Por exemplo, um classificador linear assume que 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 classificadores estatísticos freqüentemente assumem que valores de atributos estão normalmente distribuídos, e então usam os dados fornecidos para determinar média, variância e co-variância da distribuição (Michalski, Carbonell, & Mitchell, 1983).

Alguns autores têm considerado redes neurais como métodos estatísticos paramétricos uma vez que treinar uma rede neural geralmente significa encontrar valores apropriados para pesos e *bias*[1] pré-determinados.

Paradigma *instance-based*: uma forma de classificar um exemplo é lembrar de um outro similar cuja classe é conhecida e assumir que o novo exemplo terá a mesma

[1] A preferência de uma hipótese sobre outra, além da simples consistência com os exemplos, é denominada *bias* de aprendizado.

classe. Essa idéia é utilizada pelos sistemas *instance-based*, que classificam exemplos nunca vistos considerando exemplos similares conhecidos (Aha, Kibler, & Albert, 1991).

Esse tipo de sistema de aprendizado é denominado *lazy*[2]. Sistemas *lazy* necessitam manter os exemplos na memória para classificar novos exemplos. Assim, saber quais exemplos devem ser memorizados por um indutor *lazy* é muito importante. O ideal é reter apenas aqueles mais representativos do problema. *Nearest Neighbours* e Raciocínio Baseado em Casos (RBC) são, provavelmente, as técnicas mais conhecidas desse paradigma.

Paradigma conexionista: redes neurais são construções matemáticas relativamente simples que foram inspiradas no modelo biológico do sistema nervoso. Sua representação envolve unidades altamente interconectadas, da qual advém o termo conexionismo (Braga, Carvalho, & Ludermir, 2003).

A metáfora biológica com as conexões neurais do sistema nervoso tem interessado muitos pesquisadores, e tem fornecido muitas discussões sobre os méritos e as limitações dessa abordagem de aprendizado. Em particular, as analogias com a biologia têm levado muitos pesquisadores a acreditar que as redes neurais possuem um grande potencial na resolução de problemas que requerem intenso processamento sensorial humano, tal como visão e reconhecimento de voz.

Paradigma genético: esse formalismo de classificação é derivado do modelo biológico de aprendizado (Goldberg, 1989). Um classificador genético consiste de uma população de elementos de classificação que competem para fazer a predição. Elementos que possuem uma performance fraca são descartados, enquanto os elementos mais fortes proliferam, produzindo variações de si mesmos. Esse paradigma possui uma analogia direta com a teoria de Darwin, na qual sobrevivem os mais bem adaptados ao ambiente.

2.4 Aprendizado Incremental e Não Incremental

Uma outra consideração a ser feita diz respeito a forma como os exemplos são apresentados aos algoritmos de aprendizado. De acordo com esse critério os algoritmos podem ser classificados em:

1. *incrementais* e

[2] preguiçoso

2. *não incrementais.*

Algoritmos não incrementais exigem que todos os exemplos do conjunto de treinamento estejam simultaneamente disponíveis para que seja adquirido o conhecimento. Esses algoritmos devem ser utilizados no caso de todos os exemplos estarem disponíveis e os mesmos não sofrerem mudanças. A desvantagem nesse caso é que, se algum exemplo sofrer alguma alteração, ou se for necessário a inclusão de algum novo exemplo, todo o processo de indução deverá ser refeito.

Já os algoritmos incrementais modificam, se necessário, a definição do conhecimento adquirido a cada exemplo observado. A vantagem nesse caso é que o conhecimento pode ser atualizado a cada nova observação. Em alguns casos, pode ser mais eficiente revisar um conhecimento existente do que gerá-lo cada vez que um novo exemplo é observado — como ocorre com os algoritmos não incrementais.

2.5 Linguagens de Descrição

Independentemente do tipo e paradigma de aprendizado adotado, é necessário definir como o problema será traduzido em termos computacionais. Especificamente, em AM isso significa como representar exemplos, hipóteses e conhecimento de fundo (quando disponível). Em outras palavras, é necessário determinar

- a *linguagem de descrição de exemplos*,
- a *linguagem de descrição de hipóteses* e
- a *linguagem de descrição de conhecimento de fundo*,

descritas brevemente a seguir.

2.5.1 Linguagens de Descrição de Exemplos

Um exemplo, também denominado caso ou registro, é a descrição de um objeto do mundo. Na maioria dos trabalhos em AM os exemplos são descritos por um vetor de valores de atributos[3]. Entretanto, outras representações mais complexas, contendo relações entre atributos ou conjuntos de atributos do exemplo, podem ser utilizadas.

Como já mencionado, os exemplos apresentados podem estar rotulados ou não com o atributo classe. Um conjunto de exemplos rotulados com o atributo classe, geralmente, é dividido em dois conjuntos distintos: o *conjunto de treinamento*, utilizado

[3]Um atributo, também denominado característica, é uma informação que visa descrever o exemplo. Um atributo pode receber valores discretos ou contínuos.

para aprender o conceito, e o *conjunto de teste*, utilizado para avaliar o conceito aprendido. Esses dois conjuntos são normalmente disjuntos para garantir que as medidas utilizando o conjunto de teste sejam independentes e estatisticamente válidas.

Uma das maneiras mais utilizada para representação de exemplos é uma tabela no formato atributo-valor. Na Tabela 2.1 é apresentado o formato geral de uma tabela atributo-valor de um conjunto de n exemplos $E = \{E_1, E_2, \ldots, E_n\}$ e m atributos $\mathbf{X} = \{X_1, X_2, \ldots, X_m\}$. Nessa tabela, a linha i refere-se ao i-ésimo (i = 1, 2, ..., n) exemplo E, a coluna j refere-se ao j-ésimo (j = 1, 2, ..., m) atributo $\mathbf{X}$ e o valor x_{ij} refere-se ao valor do j-ésimo atributo do exemplo i. Já o valor y_i representa a classe do exemplo i. No caso de classificação, essa classe é discreta e pertence ao conjunto dos k possíveis valores de classes $\{C_1, C_2, \ldots, C_k\}$.

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

Tabela 2.1: Exemplos no formato atributo-valor

2.5.2 Linguagens de Descrição de Hipóteses

Sistemas de aprendizado supervisionado induzem uma hipótese, ou classificador, a qual pode ser descrita utilizando as seguintes linguagens de descrição:

Lógica de Predicados: também chamada de Cálculo de Predicados, é uma extensão da Lógica Proposicional que permite tratar objetos individuais e suas relações. A base da Lógica de Predicados é o predicado, que tem valor falso (F) ou verdadeiro (V), dependendo do(s) argumento(s). Como um exemplo, considere o predicado é-denso. Com o argumento chumbo, é-denso é verdadeiro (é-denso(chumbo)$\longrightarrow$V), já com o argumento algodão, é falso (é-denso(algodão)$\longrightarrow$F).

A Lógica de Predicados admite o uso de variáveis e de quantificadores (Tabela 2.2 na próxima página). A utilização de variáveis possibilita a generalização de conceitos e com os quantificadores é possível traduzir sentenças tais como:

$$\text{“Todos os gatos são mamíferos”} \quad — \quad (\forall X\, gato(X) \longrightarrow mamífero(X)) \text{ e}$$

$$\text{“Existe um gato branco”} \quad — \quad (\exists X\, gato(X) \wedge branco(X))$$

Quantificador	*Significado*
$\forall$	*paratodo*
$\exists$	*existe*

Tabela 2.2: Quantificadores utilizados na lógica de predicados

Redes Semânticas: consistem em uma estrutura composta por nós, representando objetos, conceitos ou ações, e arcos, que ligam os nós e representam a ligação, ou relação, entre eles. Um exemplo de Rede Semântica é apresentado na Figura 2.2.

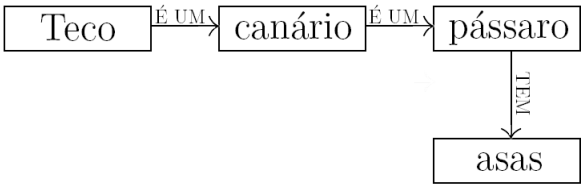


Figura 2.2: Rede semântica

Árvores de Decisão: consistem em uma estrutura em árvore sendo que, cada nó da árvore representa um atributo, cada ramo descendente desse nó corresponde a um possível valor que o atributo pode assumir e cada nó-folha da árvore está associado a uma classe. Dessa forma, um percurso na árvore (da raiz à cada nó-folha) corresponde a uma regra de classificação. Como ilustração considere o conjunto de exemplos da Tabela 2.3 com quatro exemplos, cada um representado por dois atributos binários e a classe (também binária). Na Figura 2.3 na página seguinte está ilustrada a árvore de decisão correspondente e na Figura 2.4 é ilustrado o percurso para o exemplo E_4 da Tabela 2.3. .

	A	*B*	*Classe*
E_1	0	0	0
E_2	0	1	0
E_3	1	0	0
E_4	1	1	1

Tabela 2.3: Conjunto de exemplos de treinamento

Regras de Produção: uma regra assume o formato

$$\textbf{if} < complexo > \textbf{then} < classe = C_i >$$

onde C_i pertence ao conjunto dos k possíveis valores de classes $\{C_1, C_2, ..., C_k\}$. A parte $< complexo >$ é também denominada *condições* da regra e $< classe = C_i >$ é denominada *conclusão* da regra. Um $< complexo >$ é uma disjunção

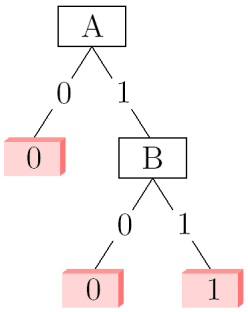


Figura 2.3: Árvore de decisão

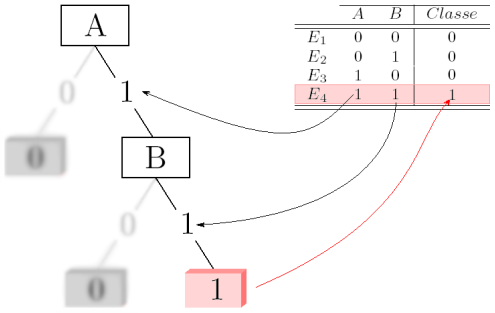


Figura 2.4: Percurso em uma árvore de decisão

de conjunções de testes em atributos do exemplo no formato X_i *op* $Valor$, onde X_i é um atributo, *op* é um operador do conjunto $\{=, \neq, <, \leq, >, \geq\}$ e $Valor$ é o valor propriamente dito do atributo. Assim, dadas as condições $<$ *complexo* $>$ a conclusão $<$ *classe* $= C_i >$ é inferida. Exemplos que satisfaçam as condições impostas pelo $<$ *complexo* $>$ compõem o conjunto de exemplos *cobertos* pela regra. Já os exemplos que satisfazem tanto as condições ($<$ *complexo* $>$) quanto a conclusão ($<$ *classe* $= C_i >$) são *positivamente cobertos* pela regra. Por outro lado, exemplos que satisfazem as condições mas possuem $<$ *classe* $\neq C_i >$ são *negativamente cobertos* pela regra, como ilustrado na Tabela 2.4.

exemplos que satisfazem...	são...
$<$ *complexo* $>$	cobertos pela regra
$<$ *complexo* $>$ **e** $<$ *classe* $= C_i >$	positivamente cobertos pela regra
$<$ *complexo* $>$ **e** $<$ *classe* $\neq C_i >$	negativamente cobertos pela regra

Tabela 2.4: Definições de cobertura da regra

2.5.3 Linguagens de Descrição de Conhecimento de Fundo

O conhecimento de fundo inclui conceitos aprendidos anteriormente, restrições do domínio, hipóteses candidatas, metas a serem inferidas, métodos para avaliar as hipóteses candidatas, etc. As possíveis linguagens de descrição de conhecimento de fundo são semelhantes às utilizadas para descrever hipóteses.

Na próxima seção, alguns conceitos relacionados com o aprendizado supervisionado, utilizados neste trabalho, são descritos com maiores detalhes.

2.6 Aprendizado Supervisionado

Como já foi mencionado, no aprendizado supervisionado o objetivo é induzir conceitos de exemplos que estão pré-classificados, *i.e.*, estão rotulados com uma classe conhecida. De acordo com os valores atribuídos à classe, o problema é conhecido como classificação ou regressão. O presente trabalho aborda o problema de classificação.

Como pode ser observado na Tabela 2.1 na página 16, exemplos rotulados são *tuplas* $E_i = (x_{i1}, x_{i2}, ..., x_{im}, y_i) = (\mathbf{x_i}, y_i)$, também referenciados por $(\mathbf{x}_i, f(\mathbf{x}_i))$, considerando que a última coluna, Y, é o atributo classe. Cada $\mathbf{x}_i$ é um elemento do conjunto $dom(X_1) \times dom(X_2) \times ... \times dom(X_m)$, no qual $dom(X_j)$ é o domínio do atributo X_j. Ou seja, $\mathbf{X}$ representa o conjunto de características e $f(\mathbf{x}_i)$ a classe a qual o exemplo pertence. Esse formato dos dados é conhecido como formato atributo-valor, apresentado na Seção 2.5 na página 15.

O processo de aprendizado se dá então pela apresentação de um conjunto de exemplos de treinamento a um indutor. A tarefa do indutor é gerar uma hipótese $\mathbf{h}$, também conhecida como *descrição de conceito*, ou classificador, que é uma aproximação da verdadeira função f, que por sua vez é desconhecida. Espera-se que, dado um novo exemplo não rotulado — $(\mathbf{x}_i, ?)$ — $\mathbf{h}$ seja capaz de predizer a sua classe — $f(\mathbf{x}_i)$.

A fim de ilustrar o processo, considere um conjunto de treinamento com quatorze exemplos, cada um descrito por quatro atributos (Tabela 2.5), apresentado em Quinlan (1986).

Nº	Atributos				Classe
	Tempo	Temperatura	Humidade	Vento	
1	ensolarado	alta	alta	fraco	não
2	ensolarado	alta	alta	forte	sim
3	nublado	alta	alta	fraco	não
4	chuvoso	moderada	alta	fraco	não
5	chuvoso	baixa	normal	fraco	não
6	chuvoso	baixa	normal	forte	sim
7	nublado	baixa	normal	forte	sim
8	ensolarado	moderada	alta	fraco	não
9	ensolarado	baixa	normal	fraco	não
10	chuvoso	moderada	normal	fraco	não
11	ensolarado	moderada	normal	forte	sim
12	nublado	moderada	alta	forte	sim
13	nublado	alta	normal	fraco	não
14	chuvoso	moderada	alta	forte	sim

Tabela 2.5: Conjunto de treinamento

Deseja-se, nesse caso, predizer se uma determinada manhã de sábado é favorável, ou não, à prática de uma determinada atividade, de acordo com as condições climáti-

cas. Cada exemplo, representando uma manhã de sábado específica, é descrito pelos atributos:

- **Tempo**, admitindo os valores *ensolarado*, *nublado* ou *chuvoso*;
- **Temperatura**, admitindo os valores *alta*, *moderada* ou *baixa*;
- **Humidade**, admitindo os valores *alta* ou *normal*;
- **Vento**, admitindo os valores *forte* ou *fraco*.

Um classificador, descrito por uma árvore de decisão que classifica corretamente cada exemplo do conjunto de treinamento, está ilustrado na Figura 2.5.

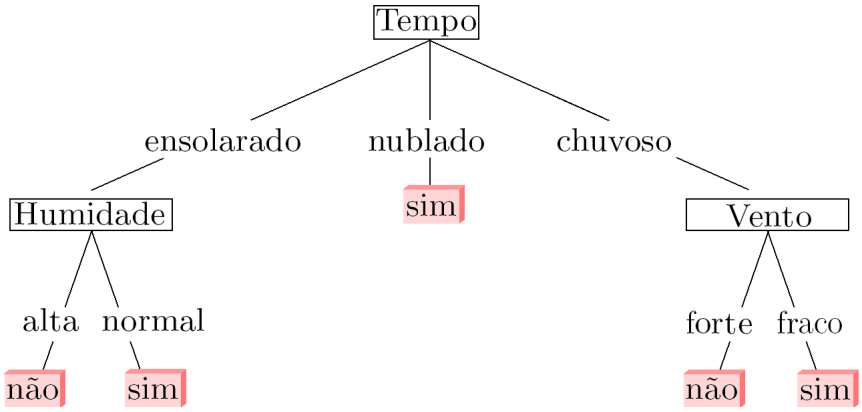


Figura 2.5: Classificador

Um ponto importante a ser observado é que várias hipóteses $\mathbf{h}$, ou várias aproximações de $f(\mathbf{X})$, podem ser obtidas a partir do mesmo conjunto de exemplos. Ou seja, a árvore de decisão ilustrada na Figura 2.5 não é necessariamente a única que classifica os exemplos da Tabela 2.5.

Na Figura 2.6 é ilustrado esse ponto. Em (a) os exemplos são representados por •. Em (b) e (c) são ilustradas hipóteses que classificam corretamente cada um desses exemplos, sendo (c) uma hipótese um pouco mais complexa que (b). Já em (d) a hipótese aparentemente ignora um dos exemplos, mas classifica corretamente os demais. A questão é qual das hipóteses escolher. Como a função f verdadeira não é conhecida, não há maneira de se escolher uma hipótese sem possuir algum conhecimento adicional.

Devido ao fato de quase sempre existir um número grande de hipóteses consistentes, todos os indutores possuem alguma forma de *bias*. De fato, aprendizado sem *bias* é impossível (Mitchell, 1982).

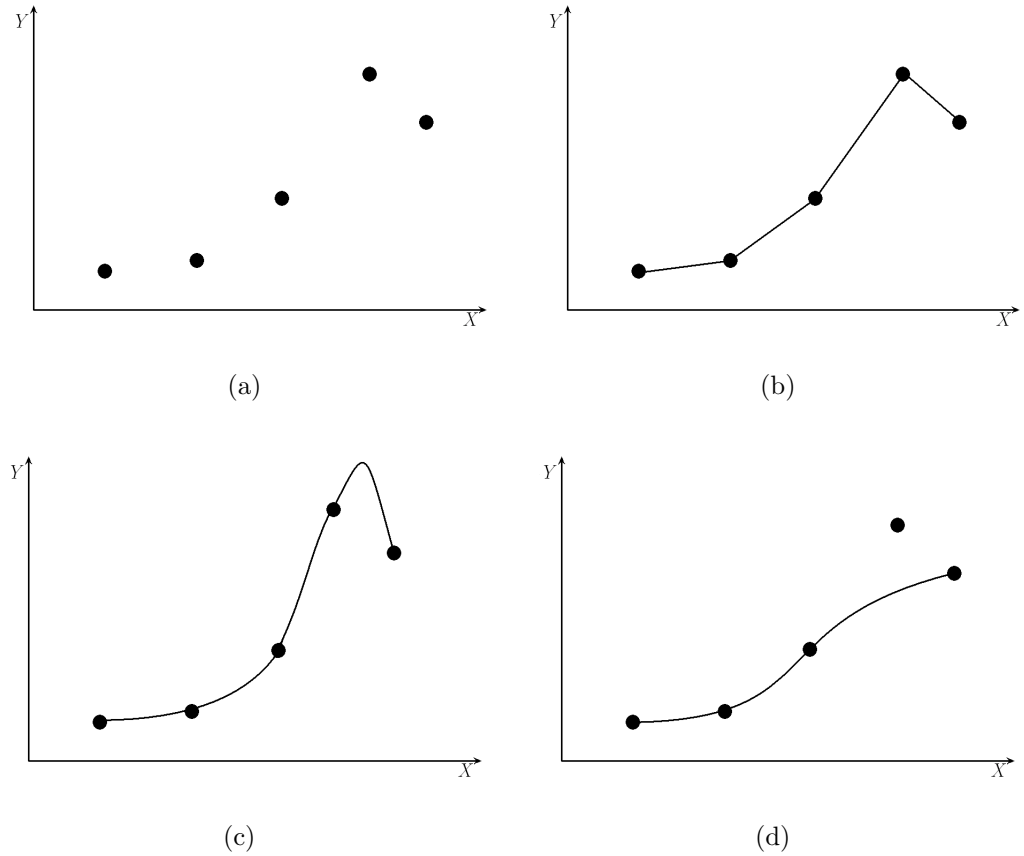


Figura 2.6: Exemplos de hipóteses que aproximam a função real desconhecida (Russel & Norvig, 2003)

2.6.1 Erro e Precisão

Como resultado do aprendizado, a hipótese é a compreensão de um conceito pelo sistema. Tal conceito é obtido pela observação dos exemplos do conjunto de treinamento. Entretanto, a principal meta do aprendizado supervisionado é generalizar esse conceito de forma a predizê-lo em exemplos não utilizados pelo indutor, *i.e.*, que não fazem parte do conjunto de treinamento. Logo, um critério importante para o sucesso do aprendizado é a precisão da hipótese em exemplos não vistos[4], ou seja, exemplos do conjunto de teste.

Uma medida muito utilizada em problemas de classificação é a *taxa de erro* do classificador $\mathbf{h}$, denotada por $ce(\mathbf{h})$. A *taxa de erro* é obtida utilizando-se a Equação 2.1 na página seguinte, a qual compara o rótulo real de cada exemplo com o rótulo predito pelo classificador. A expressão $\parallel y_i \neq \mathbf{h}(\mathbf{x}_i) \parallel$ retorna 1 caso $y_i \neq \mathbf{h}(\mathbf{x}_i)$ e zero caso

[4]É possível calcular o erro, ou a precisão, de uma hipótese utilizando o próprio conjunto de treinamento. Tais medidas são denominadas erro, ou precisão, *aparente*, mas não medem o erro em exemplos não vistos.

contrário; n é o número de exemplos. O complemento da *taxa de erro*, denominado *precisão* do classificador é dado pela Equação 2.2.

$$ce(\mathbf{h}) = \frac{1}{n}\sum_{i=1}^{n} \parallel y_i \neq \mathbf{h}(\mathbf{x_i}) \parallel \tag{2.1}$$

$$ca(\mathbf{h}) = 1 - ce(\mathbf{h}) \tag{2.2}$$

Para problemas de regressão, o erro pode ser estimado pelo cálculo da distância entre o valor real e o estimado. Geralmente, duas medidas são utilizadas: o erro quadrático médio (mse) e a distância absoluta média (mad), dados pelas Equações 2.3 e 2.4 respectivamente.

$$mse(\mathbf{h}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{h}(\mathbf{x_i}))^2 \tag{2.3}$$

$$mad(\mathbf{h}) = \frac{1}{n}\sum_{i=1}^{n}|(y_i - \mathbf{h}(\mathbf{x_i}))| \tag{2.4}$$

Freqüentemente, o cálculo da precisão ou erro de uma hipótese é realizado utilizando técnicas de amostragem[5]. Técnicas de amostragem caracterizam-se por dividir o conjunto de exemplos em várias partições, a serem utilizadas como conjuntos de treinamento e teste. A seguir são descritas algumas dessas técnicas.

Amostragem aleatória: nessa técnica de amostragem diferentes partições treinamento-teste são escolhidas de forma aleatória. Para cada partição de treinamento é induzida uma hipótese que tem então seu erro calculado a partir da partição de teste correspondente. O erro final é a média dos erros de todas as hipóteses induzidas.

***k*–*fold Cross*–*validation*:** nesse caso os exemplos são aleatoriamente divididos em k partições mutuamente exclusivas de tamanho aproximadamente igual a $\frac{n}{k}$ exemplos, sendo n o número total de exemplos. Utiliza-se então $(k-1)$ partições para treinamento, e a hipótese induzida é testada na partição restante. Esse processo é repetido k vezes, cada vez considerando uma partição diferente para teste. O erro final é a média dos erros calculados com cada uma das k partições.

***Leave*–*one*–*out*:** esta técnica é um caso especial de *k–fold cross–validation*. Nesse caso o valor de k é igual a n, *i.e.*, ao número total de exemplos. Assim, a cada iteração, um dos exemplos é utilizado para testar a hipótese induzida a partir dos

[5] *resampling*

$(n-1)$ exemplos restantes. O erro final é a soma dos erros em cada teste dividido por n. O grande problema desta técnica é o seu alto custo computacional.

2.6.2 Completude e Consistência

Uma vez induzida, uma hipótese também pode ser avaliada pela sua *completude*, *i.e.*, se ela classifica todos os exemplos, e pela sua *consistência*, *i.e.*,se ela classifica os exemplos corretamente. Assim, dada uma hipótese, ela poderá ser completa e consistente, incompleta e consistente, completa e inconsistente ou incompleta e inconsistente. Na Figura 2.7 é ilustrado um exemplo desses quatro casos considerando dois atributos X_1 e X_2 e três classes ($\circ$, $+$, $*$). As regiões azuis representam a predição da classe $\circ$ pela hipótese, regiões amarelas a predição da classe $+$ e a região vermelha a classe $*$.

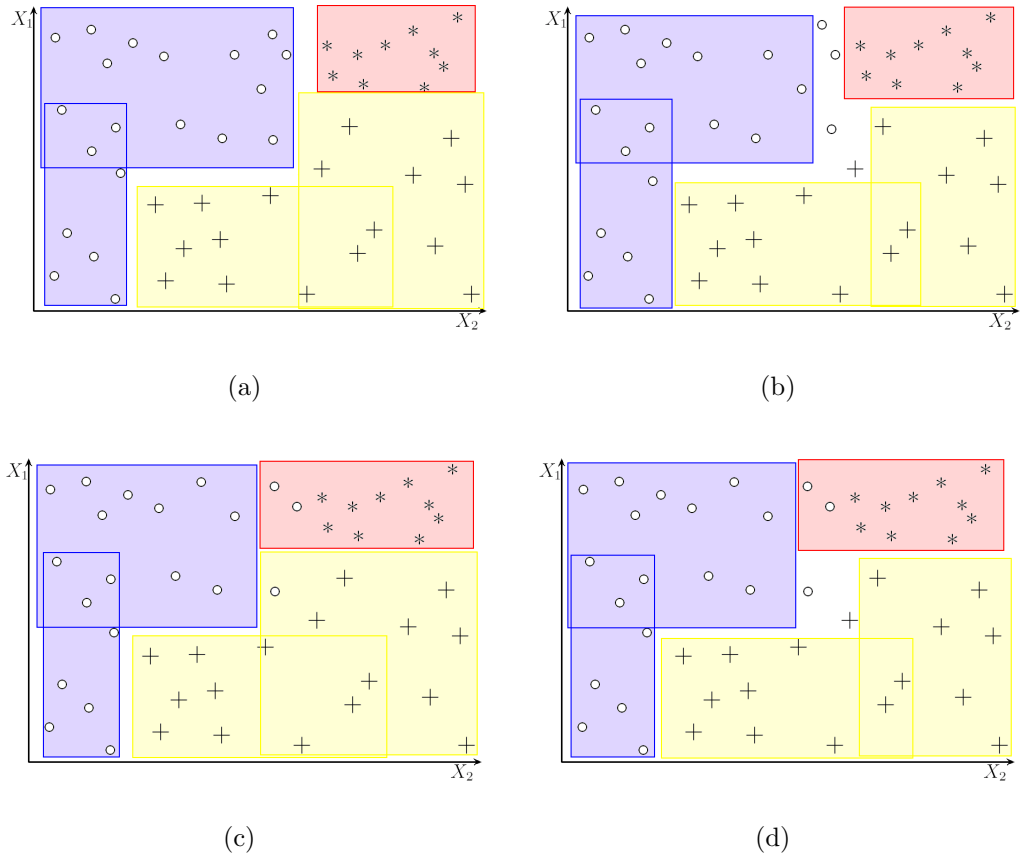


Figura 2.7: Classificação da hipótese quanto à sua completude e consistência: (a)completa e consistente, (b) incompleta e consistente, (c) completa e inconsistente, (d) incompleta e inconsistente (Monard & Baranauskas, 2003).

2.6.3 *Underfitting* e *Overfitting*

Quando um classificador é induzido, é possível que ele seja muito específico para o conjunto de treinamento, *i.e.* apresentando um alto grau de precisão para os exemplos de treinamento, e uma alta taxa de erro para o conjunto de teste. Nesse caso pode-se dizer que o classificador "decorou" os dados, não conseguindo generalizar o conceito. Essa situação é conhecida por *overfitting*.

Por outro lado, também é possível que o conjunto de treinamento seja composto por exemplos pouco representativos, ou que o usuário predefina classes muito pequenas (*e.g.*, um alto fator de poda[6] para uma árvore de decisão) ou uma combinação de ambos os casos. Nesse caso pode-se dizer que o classificador não conseguiu abstrair o conceito, apresentando baixa performance no conjunto de treinamento e no conjunto de teste. Essa situação é conhecida como *underfitting*.

2.7 Aprendizado Não Supervisionado

Por outro lado, no aprendizado não supervisionado, também conhecido como aprendizado por observação e descoberta, a tarefa do algoritmo é agrupar exemplos não rotulados, *i.e.*, exemplos que não possuem o atributo classe especificado. Nesse caso, é possível utilizar algoritmos de aprendizado para descobrir padrões nos dados a partir de alguma caracterização de regularidade, sendo esses padrões denominados clusters (Decker & Focardi, 1995; McCallum, Nigam, & Ungar, 2000). Exemplos contidos em um mesmo cluster são mais similares, segundo alguma medida de similaridade, do que aqueles contidos em clusters diferentes. O processo de formação dos clusters é geralmente conhecido por *clustering*, um dos temas tratados neste trabalho e discutido no Capítulo 3 na página 27.

2.8 Considerações Finais

Aprendizado de Máquina tem sido amplamente utilizado na tarefa de extração de conhecimento em grandes volumes de dados e suas técnicas e algoritmos estão cada vez mais presentes em processos como MD e KDD (Fayyad, Piatetsky-Shapiro, Smyth, & Uthurusamy, 1996). Neste capítulo foram apresentadas algumas linguagens de descrição utilizadas em AM, os modos que o aprendizado pode assumir e alguns conceitos

[6] Poda em árvores de decisão consiste na exclusão de algumas ramificações da árvore, tornando-a mais "simples".

básicos sobre aprendizado supervisionado, utilizado quando os exemplos estão rotulados, e não supervisionado, utilizado quandos os exemplos não estão rotulados.

No aprendizado supervisionado hipóteses são induzidas utilizando um conjunto de exemplos de treinamento. Uma vez induzida, uma hipótese pode ter sua precisão avaliada utilizando-se os exemplos do conjunto de teste. Ela também pode ser avaliada de acordo com a sua consistência e completude.

Já no aprendizado não supervisionado os padrões não são conhecidos *a priori*. Nesse caso, o objetivo é agrupar exemplos similares em clusters, de acordo com uma medida de similaridade pré-definida.

Capítulo

3

Clustering

"Falo a língua dos loucos, porque não conheço
a mórbida coerência dos lúcidos."
Luís Fernando Veríssimo

Clustering agrupa objetos (ou exemplos) em clusters[1] de modo que objetos pertencentes a um mesmo cluster são mais similares entre si de acordo com alguma medida de similaridade pré-definida, enquanto que objetos pertencentes a clusters diferentes têm uma similaridade menor. O objetivo, portanto, é maximizar a homogeneidade dos objetos dentro de um mesmo cluster enquanto maximiza–se a heterogeneidade entre objetos de clusters diferentes (Stepp III & Michalski, 1986). O processo de *clustering* compara os objetos utilizando o conjunto de atributos que os caracterizam.

Neste capítulo será dada uma visão introdutória do processo de *clustering*.

[1] agrupamentos

3.1 Introdução

A prática de classificar exemplos de acordo com a similaridade que eles apresentam é a base para a resolução de vários problemas. Segundo Jain & Dubes (1988), *clustering* é o estudo formal de algoritmos e métodos para agrupar exemplos que não estão rotulados com uma classe correspondente, o que o distingüe do aprendizado supervisionado, visto no capítulo anterior. Ou seja, os exemplos são representados como na Tabela 2.1 na página 16, mas não existe o atributo classe $\mathbf{Y}$.

Um cluster nada mais é do que um agrupamento de exemplos similares. Em Everitt (1980) são encontradas as seguintes definições de cluster:

Definição 1 *Um cluster é um conjunto de entidades que são parecidas, e entidades pertencentes a clusters diferentes não são parecidas.*

Definição 2 *Um cluster é um agrupamento de pontos no espaço de teste de tal forma que a distância entre quaisquer dois pontos em um mesmo cluster é menor que a distância entre qualquer ponto desse cluster e um outro ponto qualquer não pertencente a ele.*

Definição 3 *Clusters podem ser descritos como regiões conectadas de um espaço multi-dimensional contendo uma alta densidade relativa de pontos, separadas de outras regiões por uma região contendo uma baixa densidade relativa de pontos.*

Na Figura 3.1 são ilustrados alguns desses conceitos, onde cada exemplo é definido por dois atributos e está representado por • no espaço bidimensional. Nessa figura, com uma visão global de similaridade, pode-se perceber a existência de quatro clusters. Entretanto, com uma visão mais local, um maior número de clusters pode ser encontrado.

Qual solução é mais correta para os exemplos apresentados na Figura 3.1? Aquela com quatro ou com mais clusters? Na verdade, a solução de um processo de *clustering* é altamente dependente dos procedimentos utilizados para encontrar os clusters. De acordo com esses procedimentos, muitas soluções, *i.e.*, diferentes conjuntos de clusters, podem ser obtidas para um mesmo conjunto de exemplos. O conjunto de clusters encontrados é igualmente dependente da medida de similaridade utilizada e dos atributos dos exemplos utilizados para calcular essa medida. Inserir ou retirar atributos relevantes do conjunto de exemplos pode ter um impacto significativo na solução resultante.

Nas próximas seções são apresentados os componentes do processo de *clustering*, as medidas de similaridade, imprescindíveis nesse processo, e as técnicas de *clustering*, outro fator muito importante.

Figura 3.1: Clusters em duas dimensões (Jain & Dubes, 1988)

3.2 Componentes do Processo de Clustering

Uma atividade de *clustering* é composta de algumas etapas seqüenciais. Três etapas primordiais são mostradas graficamente na Figura 3.2 e descritas a seguir.

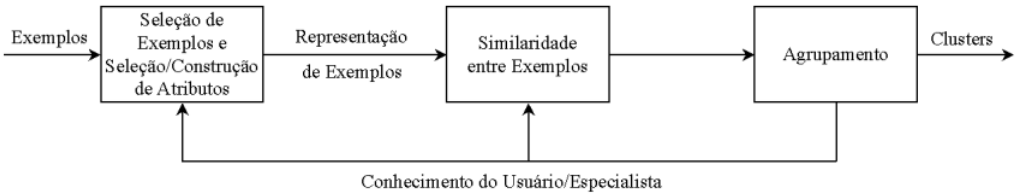


Figura 3.2: Etapas primordiais da tarefa de *clustering* (Jain, Murty, & Flynn, 1999)

1. Seleção de Exemplos e Seleção/Construção de Atributos: refere-se ao número de exemplos a serem considerados e a seleção de atributos relevantes e/ou construção de novos atributos representativos em função dos atributos originais. Existem duas abordagens principais para seleção/construção de atributos: Seleção de um Subconjunto de Atributos (SSA) e Indução Construtiva (IC) . No primeiro caso, dado um conjunto de m atributos, após o processo de SSA, será obtido um subconjunto de atributos de tamanho m_{SSA} tal que $m_{SSA} < m$. Ou seja, SSA produz uma diminuição do número de atributos que serão posteriormente considerados no processo de indução (Baranauskas & Monard, 1998, 1999; Blum & Langley, 1997; John, Kohavi, & Pfleger, 1994; Lee, Monard, & Baranauskas, 1999).

No caso de Indução Construtiva, dado o mesmo conjunto de atributos de tamanho m, combinando os atributos originais, ditos atributos primitivos, obtém-se novos atributos que, possivelmente, podem melhorar a linguagem de descrição do conjunto de exemplos (Flach & Lavrač, 2000; Lee, 2000; Lee & Monard, 2000). Desse modo, diferente da SSA, a IC aumenta o número de atributos, *i.e.*, ao final desse processo serão obtidos m_{IC} atributos, sendo $m_{IC} > m$;

2. Similaridade entre Exemplos: refere-se à seleção da medida de similaridade a ser utilizada, a qual deve ser apropriada ao domínio dos dados, para encontrar a proximidade entre exemplos. Uma medida simples como a distância Euclideana é freqüentemente utilizada nessa etapa;

3. Agrupamento: refere-se ao processo de agrupamento, que é a técnica utilizada para agrupar os exemplos em clusters. Existe uma variedade de técnicas de agrupamento e a representação final dos clusters depende da técnica utilizada.

3.3 Medidas de Similaridade

Uma vez que *clustering* consiste em agrupar exemplos de tal forma que exemplos pertencentes a um mesmo cluster sejam mais similares entre si, de acordo com alguma medida de similaridade pré-definida, do que exemplos pertencentes a clusters diferentes, torna-se necessário e extremamente importante definir *a priori* o conceito de similaridade utilizado nesse processo.

A similaridade entre exemplos pode ser calculada de diversas maneiras, mas três métodos podem ser destacados em aplicações de *clustering*:

1. *medidas de distância*,

2. *medidas de correlação* e

3. *medidas de associação*.

Cada um dos métodos representa uma perspectiva de similaridade, dependendo dos objetivos e do tipo dos dados. As medidas de distância e de correlação requerem dados contínuos, enquanto que as medidas de associação são utilizadas para dados discretos. Estas medidas são sucintamente descritas a seguir.

3.3.1 Medidas de Distância

Considerando os atributos dos exemplos como dimensões de um espaço multidimensional, a descrição de cada exemplo corresponderá a um ponto no espaço. Assim, pode-se definir a similaridade entre dois exemplos como sendo a distância entre eles. Isto é, caso dois exemplos sejam similares, então a distância entre eles é pequena. Conforme a similaridade diminui, a distância aumenta.

Não há uma única maneira de se definir distância. Por exemplo, para encontrar a distância entre dois pontos em um plano, basta encontrar o tamanho da linha reta que os une. Já no caso de encontrar a distância entre duas cidades, como São Carlos e São Paulo, por exemplo, uma linha reta não seria adequada, uma vez que ela atravessaria o solo, ao invés de acompanhar a superfície da Terra. Nesse caso, uma medida de distância mais adequada seria encontrar o tamanho da curva que liga as duas cidades.

Um outro exemplo é a medida conhecida por distância ferrovia francesa. Ela é baseada no fato que (ao menos no passado) a maioria das ferrovias da França tinham Paris como destino. Assim, a menor distância entre dois pontos era a simples distância entre eles, caso estivessem ligados pela mesma ferrovia, ou então a soma entre a distância do primeiro ponto até Paris e a distância entre Paris e o segundo ponto.

Como foi ilustrado, há várias maneiras de se calcular a distância entre dois pontos $\mathbf{x}$ e $\mathbf{y}$. Entretanto, para uma medida D ser considerada medida de distância, ela deve satisfazer as seguintes condições:

1. $D(\mathbf{x}, \mathbf{y}) >= 0$[2]
2. $D(\mathbf{x}, \mathbf{y}) = 0$ se e somente se $\mathbf{x} = \mathbf{y}$[3]
3. $D(\mathbf{x}, \mathbf{y}) = D(\mathbf{y}, \mathbf{x})$[4]
4. $D(\mathbf{x}, \mathbf{y}) <= D(\mathbf{x}, \mathbf{z}) + D(\mathbf{z}, \mathbf{y})$[5]

A seguir são mostrados alguns exemplos de medidas de distância, onde $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ e $\mathbf{y} = (y_1, y_2, ..., y_m)$ representam os valores dos atributos $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_m$ desses exemplos.

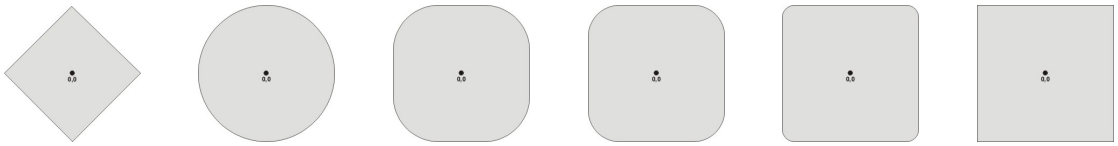
[2] A distância entre dois pontos não pode ser negativa.
[3] A distância entre qualquer ponto e ele mesmo é zero.
[4] A distância entre $\mathbf{x}$ e $\mathbf{y}$ é igual a distância entre $\mathbf{y}$ e $\mathbf{x}$.
[5] Desigualdade triangular.

Minkowsky

A medida de distância Minkowsky, definida pela Equação 3.1, estabelece uma forma genérica para calcular a distância entre dois pontos de acordo com o valor de r utilizado.

$$D(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{m} |x_i - y_i|^r \right)^{\frac{1}{r}} \tag{3.1}$$

Na Figura 3.3 é ilustrada a região formada pelos pontos igualmente distantes da origem segundo a distância Minkowski para vários valores de r.



(a) (b) (c) (d) (e) (f)

Figura 3.3: Efeito da variação de r na medida Minkowski. (a)$r = 1$ (b)$r = 2$ (c)$r = 3$ (d)$r = 4$ (e)$r = 20$ (f)$r = \infty$

Quando $r = 1$, a medida Minkowski é conhecida como medida Manhattan-city/block (Equação 3.2), e quando $r = 2$, ela é conhecida como medida Euclideana (Equação 3.3). Conforme o valor de r aumenta, os pontos eqüidistantes de um ponto central estão situados em um quadrado. Para $r = \infty$, esses pontos estão situados em um quadrado perfeito.

Manhattan/city-block

Esta medida, definida pela Equação 3.2, é simplesmente a soma das diferenças absolutas das dimensões.

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} (|x_i - y_i|) \tag{3.2}$$

É assim chamada pois em várias cidades é praticamente impossível estabelecer uma rota entre dois pontos através de uma reta, devido ao fato das cidades serem freqüentemente subdivididas em quadras, e estas não estarem vazias, mas sim preenchidas com prédios, por exemplo. Assim, qualquer rota entre dois pontos é parecida com a ilustrada na Figura 3.4.

Na Figura 3.5 é ilustrada a região formada pelos pontos igualmente distantes da origem segundo a distância Manhattan/city-block.

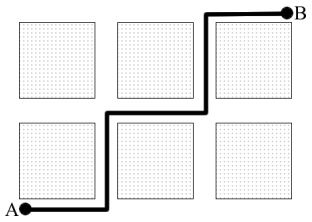


Figura 3.4: Distância Manhattan

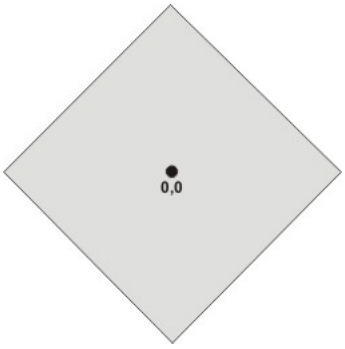


Figura 3.5: Formato do cluster encontrado pela distância Manhattan/city-block

Euclideana

A medida Euclideana, definida pela Equação 3.3, é provavelmente a medida de distância mais utilizada. Ela expressa a distância geométrica Euclideana entre os exemplos em um espaço multidimensional.

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \tag{3.3}$$

Na Figura 3.6 é ilustrada a região formada pelos pontos igualmente distantes da origem segundo a distância Euclideana.

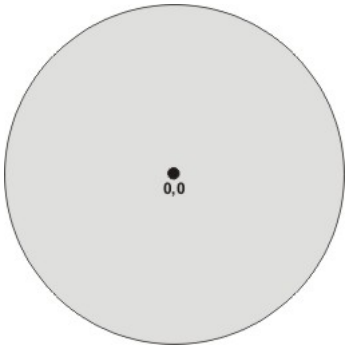


Figura 3.6: Formato do cluster encontrado pela distância Euclideana

Uma alternativa à medida Euclideana é a medida Euclideana Quadrada, definida

pela Equação 3.4, semelhante à anterior mas sem extrair a raiz quadrada. A grande vantagem dessa medida é a diminuição do tempo computacional para calculá-la.

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} (x_i - y_i)^2 \tag{3.4}$$

Camberra

A medida Camberra é definida pela Equação 3.5.

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|} \tag{3.5}$$

Chebychev

Essa medida, definida pela Equação 3.6 é a máxima diferença absoluta entre as dimensões. Ela é apropriada para casos nos quais dois exemplos são considerados diferentes caso possuam qualquer atributo diferente.

$$D(\mathbf{x}, \mathbf{y}) = \max_{i=1}^{m} |x_i - y_i| \tag{3.6}$$

Na Figura 3.7 é ilustrada a região formada pelos pontos igualmente distantes da origem segundo a distância Chebychev.

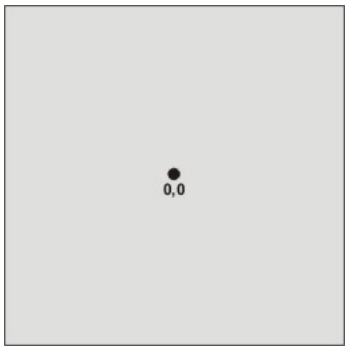


Figura 3.7: Formato do cluster encontrado pela distância Chebychev

Mahalanobis

A distância de Mahalanobis é definida pela Equação 3.7, onde V é a matriz de covariância dos atributos $X_1, X_2, X_3, ..., X_m$, e $detV$ é o determinante dessa matriz.

$$D(\mathbf{x}, \mathbf{y}) = [detV]^{\frac{1}{m}} (\mathbf{x} - \mathbf{y})^T V^{-1} (\mathbf{x} - \mathbf{y}) \tag{3.7}$$

Na Figura 3.8 é ilustrado um exemplo de região formada pelos pontos igualmente distantes da origem segundo a distância Mahalanobis. Entretanto, dependendo dos dados, essa medida pode originar clusters com formatos elipsoidais com rotação à direita, e até mesmo clusters circulares.

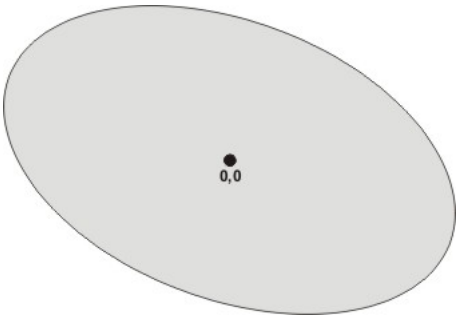


Figura 3.8: Formato do cluster encontrado pela distância Mahalanobis

Chi-quadrado

Essa medida, definida pela Equação 3.8, onde sum_i é a soma de todos os valores do atributo i do conjunto de exemplos, e $size_x$ e $size_y$ é a soma de todos os valores dos atributos de $\mathbf{x}$ e $\mathbf{y}$ respectivamente, é baseada no teste Chi-quadrado de igualdade para dois conjuntos de freqüências.

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} \frac{1}{sum_i} \left(\frac{x_i}{size_x} - \frac{y_i}{size_y} \right)^2 \tag{3.8}$$

Qual medida de distância escolher?

Não existe uma regra geral que defina qual medida de distância deve ser utilizada em determinada situação. A escolha da medida ideal para um determinado problema se dá geralmente após a realização de vários testes com diferentes medidas de distâncias. Entretanto, uma ressalva pode ser feita caso seja conhecido o formato do cluster esperado. Nesse caso a distância mais apropriada será aquela que apresente, para pontos eqüidistantes da origem, um formato idêntico, ou parecido com aquele procurado. Na Figura 3.9 é apresentada uma comparação entre os formatos dos clusters encontrados para algumas das medidas de distância apresentadas.

3.3.2 Medidas de Correlação

Uma medida de similaridade entre exemplos pode ser o coeficiente de correlação entre um par de exemplos, o qual é medido utilizando diversos atributos. Medidas

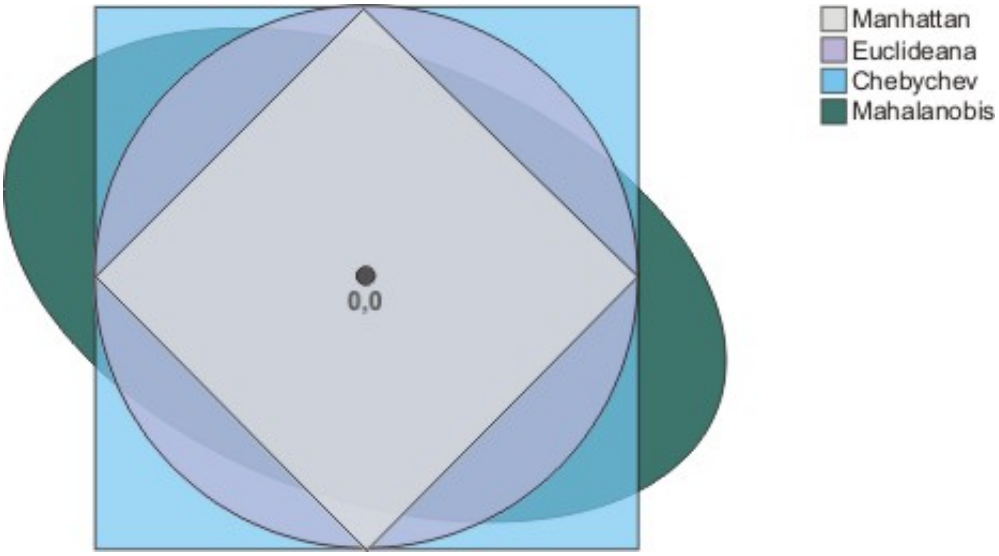


Figura 3.9: Comparação entre os formatos dos clusters

de correlação representam similaridade pela correspondência de exemplos por meio de suas características

A medida de correlação não considera a magnitude, mas o padrão nos valores dos atributos. Ela pode ser calculada pela Equação 3.9.

$$D(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{m} (x_i - \bar{x_i})(y_i - \bar{y_i})}{\sqrt{\sum_{i=1}^{m} (x_i - \bar{x_i})^2 \sum_{i=1}^{m} (y_i - \bar{y_i})^2}} \tag{3.9}$$

3.3.3 *Medidas de Associação*

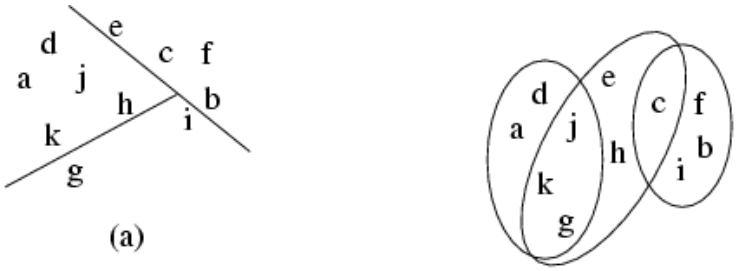
Medidas de associação são utilizadas para comparar exemplos cujas características são medidas apenas para atributos com valores discretos. Por exemplo, valores de atributos para uma resposta poderiam ser “sim” ou “não”, e uma medida de associação poderia verificar o grau de concordância entre cada par de exemplos. A forma mais simples de medida de associação seria considerar a percentagem de vezes que ocorrer uma concordância (ambos casos responderam “sim” ou ambos responderam “não”) para valores dos atributos. Extensões desse coeficiente simples de concordância têm sido desenvolvidos para acomodar atributos nominais multi-categóricos e medidas ordinais niveladas. Uma revisão de vários tipos de medidas de associação pode ser encontrada em Everitt (1980) e Sneath & Sokal (1973).

3.4 Técnicas de Clustering

Não existe nenhuma técnica de *clustering* universalmente aplicável em qualquer conjunto de dados. É necessário, portanto, avaliar as diferentes técnicas existentes para escolher a que melhor se adeque a uma aplicação particular. As diferentes técnicas de *clustering* são classificadas de acordo com os resultados obtidos pelos algoritmos em quatro grandes grupos:

1. *técnicas de otimização*,
2. *técnicas de grupo*,
3. *técnicas probabilísticas* e
4. *técnicas hierárquicas*.

Na Figura 3.10 é sumarizada a classificação dessas técnicas, as quais são descritas a seguir (Witten & Frank, 2000; Chiara, 2003).

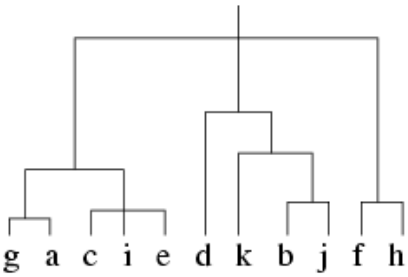


(a)

(b)

	1	2	3
a	0.4	0.1	0.5
b	0.1	0.8	0.1
c	0.3	0.3	0.4
d	0.1	0.1	0.8
e	0.4	0.2	0.4
f	0.1	0.4	0.5
g	0.7	0.2	0.1
h	0.5	0.4	0.1

(c)



(d)

Figura 3.10: Técnicas de *clustering*: (a) otimização (b) *clumping* (c) probabilística (d) hierárquica

Técnicas de otimização: visam formar uma k-partição ótima sobre os exemplos, *i.e.*, divide o conjunto em k clusters mutuamente exclusivos, sendo o valor de k dado pelo usuário/especialista. As técnicas de otimização são computacionalmente caras, pois fazem uma busca exaustiva pela k-partição ótima. Por isso, seu uso é restrito a pequenas quantidades de exemplos e/ou pequenos valores de k.

Técnicas de grupo (*clumping*): retornam clusters nos quais seus elementos (exemplos) podem se sobrepor. O problema com essas técnicas é que um mesmo exemplo pode pertencer a vários clusters.

Técnicas probabilísticas: associam, para cada exemplo, a probabilidade dele estar em cada cluster.

Técnicas hierárquicas: criam árvores comumente chamadas de dendrogramas. As folhas da árvore representam exemplos individuais e os nós internos representam grupos (clusters) de exemplos. Esses métodos podem ser subdivididos em aglomerativos e divisivos. Os métodos aglomerativos constroem a árvore de baixo para cima (das folhas, ou exemplos, para a raiz), enquanto os divisivos constroem a árvore de cima para baixo (da raiz para as folhas). As técnicas hierárquicas são computacionalmente mais baratas que as técnicas de otimização.

Deve ser observado que esta não é a única maneira de classificar métodos de *clustering*. Outras classificações podem ser encontradas em Michalski & Stepp (1992) e Jain et al. (1999).

3.5 Algoritmos de Clustering

Muitos algoritmos de *clustering* têm sido propostos. Nessa seção são descritos alguns dos algoritmos de *clustering* mais conhecidos e utilizados pela comunidade.

3.5.1 *k-means*

O algoritmo *k-means* (Macqueen, 1967), que utiliza a técnica de otimização, é um dos algoritmos mais freqüentemente utilizado. Como entrada, o usuário/especialista deve especificar o número k de partições.

Inicialmente, o algoritmo escolhe, aleatoriamente, k pontos que serão os centros dos clusters iniciais — centróides. O próximo passo é determinar, para cada exemplo do conjunto de dados, o cluster ao qual ele pertence. Isso é feito calculando a distância entre o exemplo e o centro de cada cluster. O exemplo irá pertencer ao cluster do

qual ele estiver mais perto do centro. Depois, é calculado um novo centróide para cada cluster, ou seja, os pontos iniciais não são os centros definitivos dos clusters, eles são apenas uma tentativa inicial. O centróide calculado, para cada cluster, passa a ser o novo centro. O processo se repete até que os centros dos clusters se estabilizem, *i.e.*, o mesmo ponto é escolhido como centro durante algumas iterações.

Porém, esse algoritmo apresenta alguns problemas. Os centros dos clusters finais nem sempre representam uma solução ótima. Isso porque diferentes configurações de clusters podem ser obtidas a partir de diferentes escolhas para os centros dos clusters iniciais. Na Figura 3.11 é ilustrada uma situação em que esse problema ocorre.

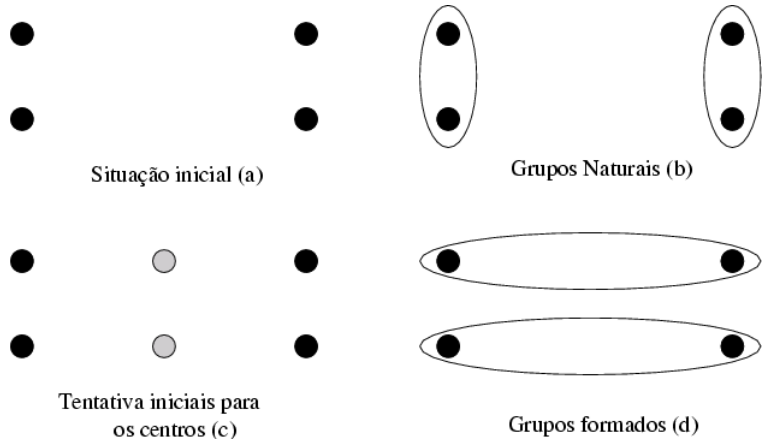


Figura 3.11: Um dos problemas do algoritmo *k-means*

Nessa situação, existem quatro pontos (exemplos) que formam um retângulo (a). Os agrupamentos naturais seriam os dois formados pelos pontos dos lados menores do retângulo (b). Porém, supondo que os centros iniciais caiam no meio dos lados maiores (c), essa configuração forma uma situação estável, fazendo com que os clusters finais sejam os dois formados pelos pontos dos lados maiores do retângulo (d), ou seja, os clusters formados pelo algoritmo podem não ser os naturais.

Deve ser observado que o resultado mostrado não está errado mas, para os seres humanos, não é o agrupamento natural. Entretanto, caso os centros iniciais escolhidos fossem outros, os clusters naturais poderiam ser formados. Assim, para encontrar os melhores agrupamentos, ou não convergir para um mínimo local, é necessário executar o algoritmo várias vezes, com diferentes centros iniciais, e escolher o melhor resultado.

Como uma outra alternativa, há alguns estudos no intuito de se encontrar métodos para a seleção dos centros iniciais dos clusters, e não mais escolhê-los aleatoriamente (Bradley & Fayyad, 1998; Hamerly & Elkan, 2002).

3.5.2 EM

O fato de, normalmente, não haverem evidências suficientes sobre os dados de forma a se tomar uma decisão determinística sobre qual exemplo deve pertencer a qual cluster, mostra a necessidade de se especificar a probabilidade dos exemplos de pertencer a um cluster ou outro. O algoritmo EM (*Expectation-Maximization*) faz justamente isso.

O algoritmo EM , assim como a maioria dos algoritmos que pertencem à classe das técnicas probabilísticas, baseia-se em um modelo estatístico chamado "misturas finitas". Uma mistura finita é um conjunto de k distribuições de probabilidades, representando k clusters, que governam os valores dos atributos dos exemplos naquele cluster. Em outras palavras, cada distribuição determina a probabilidade de um determinado exemplo possuir um certo conjunto de valores de atributos dado que esse exemplo pertence a esse cluster. Cada cluster tem uma distribuição diferente sendo que, na realidade, cada um dos exemplos pertence a um e somente um desses clusters, mas não se sabe a qual deles.

O exemplo mais simples do modelo de misturas finitas ocorre quando os exemplos consistem de apenas um atributo numérico, que tem uma distribuição normal para cada cluster, mas com diferentes médias e variâncias. O problema de *clustering* resume-se a considerar um conjunto de exemplos, que nesse caso trata-se simplesmente de exemplos com apenas um valor numérico, um número pré-especificado k de clusters, e calcular a média e a variância de cada cluster, bem como a distribuição da população entre os clusters.

Por exemplo, considere dois clusters A e B cada um com uma distribuição normal com média μ_A e desvio padrão σ_A para A e μ_B e σ_B para B, ilustrados na Figura 3.12 na próxima página. Exemplos são amostrados dessas distribuições, usando as probabilidades ρ_A para A e ρ_B para B ($\rho_A + \rho_B = 1$), resultando em um conjunto de dados formados pela classe de onde ele veio e o valor que ele assume — Figura 3.12.

Suponha, agora, que seja fornecido o conjunto de dados sem o cluster ao qual eles pertencem e deseja-se determinar os seis parâmetros (na realidade cinco parâmetros já que $\rho_A + \rho_B = 1$) que caracterizam o modelo (μ_A, σ_A, μ_B, σ_B, ρ_A, ρ_B). Esse é problema da mistura finita. Se fosse conhecido de qual dessas duas distribuições provem cada exemplo, então, é simples encontrar os parâmetros μ_A, σ_A, μ_B, σ_B, do modelo estimado e, conseqüentemente, a média e o desvio padrão para as amostras de A e B por meio

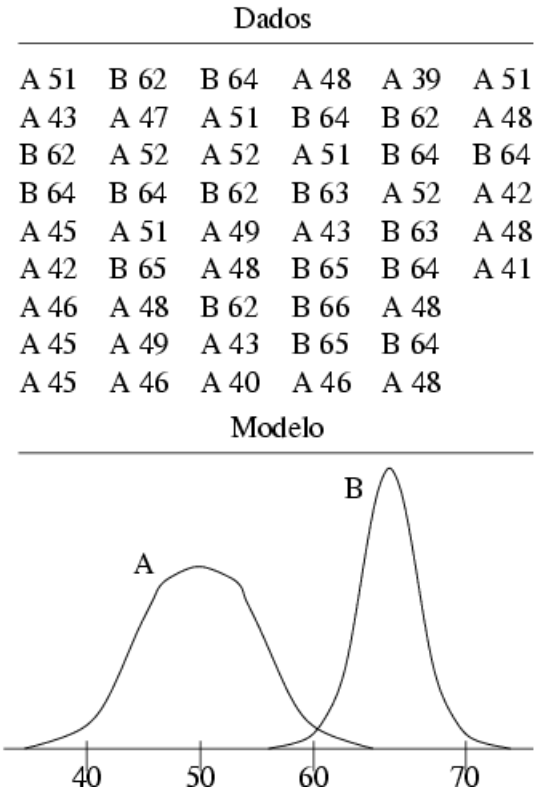


Figura 3.12: Um modelo de mistura com duas classes: $\mu_A = 50, \sigma_A = 5, \rho_A = 0.6; \mu_B = 65, \sigma_B = 2, \rho_B = 0.4$ (Witten & Frank, 2000)

das Equações 3.10 e 3.11[6] .

$$\mu = \frac{x_1 + x_2 + ... + x_n}{n} \tag{3.10}$$

$$\sigma = \sqrt{\frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + ... + (x_n - \mu)^2}{n - 1}} \tag{3.11}$$

O quinto parâmetro ρ_A pode ser estimado considerando a proporção de exemplos no cluster A. Conhecidos esses cinco parâmetros, lembrando que $\rho_B = 1 - \rho_A$, é simples encontrar a probabilidade que um dado exemplo provem de cada distribuição já que, dado um exemplo x, a probabilidade de x pertencer ao cluster A é dada pela Equação 3.12, onde $f(x; \mu_A, \sigma_A)$ é a distribuição normal para o cluster A, definida pela Equação 3.13.

$$Pr(A|x) = \frac{Pr(x|A).Pr(A)}{Pr(x)} = \frac{f(x; \mu_A, \sigma_A)\rho_A}{Pr(x)} \tag{3.12}$$

[6]O uso de $(n-1)$ no denominador da Equação 3.11 é uma particularidade de amostragem pois, na prática, há pouca diferença se n for utilizado em lugar de $n-1$.

$$f(x;\mu_A,\sigma_A) = \frac{1}{\sqrt{2\pi}\sigma_A} e^{\frac{-(x-\mu_A)^2}{2{\sigma_A}^2}} \tag{3.13}$$

O problema é que não se sabe nem de quais clusters vieram os exemplos e nem os parâmetros do modelo de mistura. O que o algoritmo EM faz é utilizar o mesmo procedimento utilizado no algoritmo *k-means*: faz-se uma tentativa inicial para estimar os parâmetros do modelo, utilizam-se esses parâmetros para calcular as probabilidades dos exemplos pertencerem a cada cluster (*expectation*), reestimam-se os parâmetros com base nas probabilidades calculadas (*maximization*) e repete-se o processo.

Como mencionado na Subseção 3.5.1 na página 38, o algoritmo *k-means* finaliza quando os centros dos clusters se estabilizam. Já o algoritmo EM tem um problema: ele converge para um valor fixo, mas nunca o atinge. Para contornar esse problema, o algoritmo calcula um valor chamado probabilidade geral. Esse valor aumenta a cada iteração e mede a qualidade dos agrupamentos. Com isso, o processo de iteração se repete até que o aumento da probabilidade geral se torne desprezível. Deve ser ressaltado que o algoritmo EM não calcula os agrupamentos ótimos, sendo que todo o processo deve ser repetido algumas vezes para se obter o melhor resultado. Em McLachlan & Krishnan (1997) encontra-se uma explicação detalhada do algoritmo EM e algumas extensões propostas.

3.5.3 *Algoritmo Incremental*

Esse algoritmo trabalha individualmente com cada exemplo do conjunto de dados formando uma árvore, na qual cada folha é um exemplo e a raiz representa todo o conjunto de exemplos, ou seja, o algoritmo pertence à classe das técnicas hierárquicas.

O algoritmo utiliza o método divisivo (Seção 3.4 na página 37). Assim, inicialmente, a árvore é formada apenas pela raiz. Os exemplos são adicionados, um de cada vez, formando um novo cluster ou juntando-se a um cluster já existente. O algoritmo também realiza reestruturações maiores na árvore. Para decidir onde cada exemplo será colocado, um valor denominado "utilidade da categoria" é calculado. Esse valor mede a qualidade de um agrupamento. Cada novo exemplo é processado tentando colocá-lo em cada um dos nós filhos da raiz calculando a utilidade da categoria para cada possibilidade de agrupamento. Se não for oportuno, o exemplo é colocado como um novo filho da raiz. Mas, se a utilidade da categoria for boa, então o exemplo forma um novo cluster com aquele nó filho. Caso esse nó já seja um cluster, o algoritmo é chamado recursivamente com esse cluster sendo a raiz da sub-árvore. Na Figura 3.13 na próxima página são ilustrados alguns passos desse algoritmo.

Porém, se o algoritmo seguisse sempre esses passos, a estrutura da árvore seria

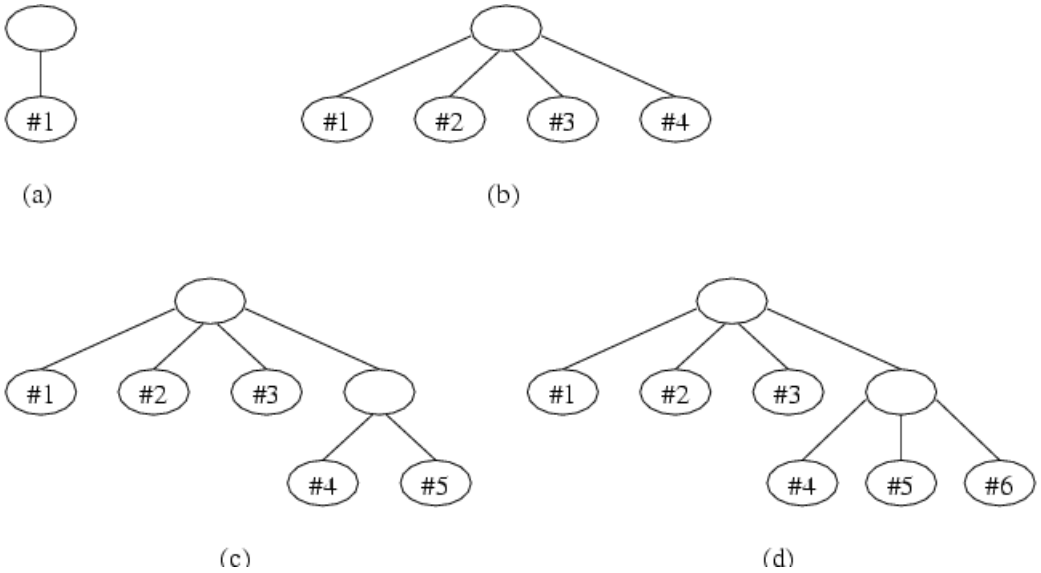


Figura 3.13: Passos do algoritmo incremental: (a) situação inicial (b) alguns exemplos são adicionados, mas não formam clusters (c) um primeiro cluster é formado (d) mais um exemplo é adicionado ao novo cluster (Witten & Frank, 2000)

fortemente dependente da ordem de processamento dos exemplos. Para contornar esse problema, o algoritmo, ao calcular a utilidade da categoria para cada um dos nós, armazena os dois melhores valores. O melhor valor é utilizado para formar um novo cluster a menos que seja melhor criar um novo cluster separado para o novo exemplo. No entanto, antes de formar o cluster, o algoritmo verifica a possibilidade de agrupar o segundo melhor valor com o primeiro. Na Figura 3.14 é ilustrada essa etapa.

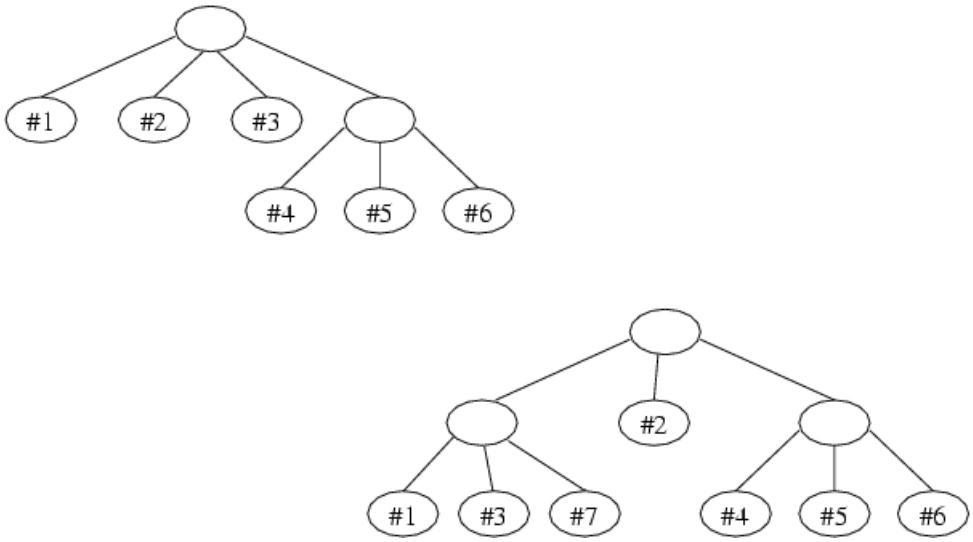


Figura 3.14: Reestruturação da árvore (Witten & Frank, 2000)

A operação inversa também pode ser executada, ou seja, a divisão de um cluster.

Sempre que o melhor valor para a utilidade da categoria é encontrado e a união de dois clusters (melhor valor e segundo melhor valor) não for indicada, o algoritmo verifica a possibilidade de dividir o cluster de melhor valor. As operações de união e divisão proporcionam uma maneira de compensar escolhas erradas causadas pela ordem de processamento dos exemplos.

Um problema que ocorre com esse algoritmo é que ele cria uma árvore de hierarquia sobrecarregada, com cada exemplo do conjunto de dados em suas folhas. Para evitar isso, um parâmetro de corte é dado para evitar o crescimento da árvore. O corte é especificado em termos da função de utilidade da categoria. Sempre que o aumento na utilidade da categoria, causado pela adição de um novo nó, for muito pequeno, esse nó não é considerado.

3.6 *Clusters versus Classes*

É importante observar que dois ou mais clusters podem agrupar exemplos que referem-se a uma mesma instância de um determinado conceito (a classe do exemplo em aprendizado supervisionado). Isto é ilustrado claramente na Figura 3.15 na próxima página.

O caso (a) ilustra um conjunto de exemplos de treinamento rotulados com a classe "+" e "−". Após submeter esses exemplos a um algoritmo de aprendizado supervisionado que induz o conceito utilizando uma árvore de decisão, por exemplo, as regras induzidas seriam do tipo

$$\textbf{if } x < a \textbf{ and } y < b \textbf{ then} \text{ classe "-"}$$
$$\textbf{if } x \geq a \textbf{ and } y \geq b \textbf{ then} \text{ classe "-"}$$
$$\textbf{if } x < a \textbf{ and } y \geq b \textbf{ then} \text{ classe "+"}$$
$$\textbf{if } x \geq a \textbf{ and } y < b \textbf{ then} \text{ classe "+"}$$

Entretanto, no caso de desconhecer a classe, os exemplos de treinamento são vistos pelo algoritmo de *clustering* como mostrado pela Figura 3.15(b), *i.e.*, apenas como pontos no espaço de busca que podem ser agrupados de acordo com algum critério de similaridade.

Nesse caso, o algoritmo de *clustering* encontraria 4 (quatro) clusters distintos — $C(1)$, $C(2)$, $C(3)$ e $C(4)$ (Figura 3.15(c)). Porém, esse conjunto de treinamento representa apenas duas instâncias de um determinado conceito, representadas pelas classes "+" (clusters 1 e 4) e "−" (clusters 2 e 3).

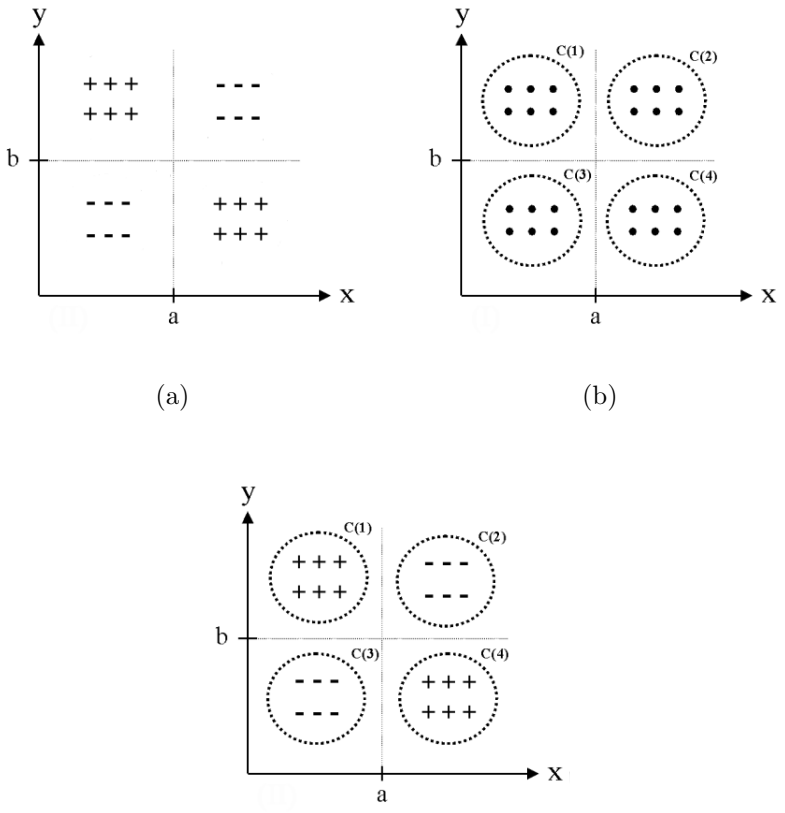


Figura 3.15: Clusters *versus* classes: (a) conjunto de treinamento (b) clusters encontrados (c) clusters diferentes expressando o mesmo conceito (mesma classe) (Martins, Monard, Haedo, & Matsudo, 2001)

Evidentemente, seria interessante, e desejado, que todos os exemplos representando a mesma instância de um conceito, ou a mesma classe, fossem agrupados em um mesmo cluster. Infelizmente, isso não é garantido em um processo de *clustering*.

Para se atingir tal objetivo faz-se necessária uma interpretação, por um especialista, nos clusters encontrados, a fim de se tentar extrair o significado conceitual de cada cluster e identificar possíveis "fusões". Na Figura 3.15(c) as fusões "desejáveis" seriam entre os clusters $C(1)$ e $C(4)$ e entre $C(2)$ e $C(3)$. Entretanto, interpretar clusters não é uma tarefa trivial. De fato, isso pode ser mais difícil do que a própria geração dos clusters.

Como uma alternativa a esse problema, encontra-se em desenvolvimento no LABIC um projeto de doutorado que visa a criação de uma metodologia para auxiliar o especialista no processo de interpretação de clusters (Martins & Monard, 2000; Martins, Monard, Haedo, & Matsudo, 2001; Martins, Monard, & Halembeck, 2002b,a).

3.7 Alguns Sistemas de Clustering

Na Tabela 3.1 são apresentadas algumas informações de sistemas freqüentemente citados na literatura que realizam, entre outros, tarefas de *clustering* (Halembeck, 2001). Vários desses sistemas possuem ferramentas gráficas de visualização.

Sistema	Técnica	Software Livre	Linguagem
Autoclass	probabilística	sim	Lisp/C
	http://www.ic-www.arc.nasa.gov/ic//projects/bayes-group/autoclass		
Clustan Graphics	hierárquica	não	C
	http://www.clustan.com/clustangraphics_features.html		
CViz	probabilística	sim	Java
	http://www.alphaworks.ibm.com/tech/cviz		
DataMiner	otimização	não	Java
	http://www.data-miner.com		
EMclust	hierárquica	sim	Fortran
	http://www.stat.washington.edu/fraley/mclust/soft.shtml		
Mineset	otimização	não	C++
	http://www.sgi.com/software/mineset		
Normix	probabilística	sim	Fortran
	http://www.alumnus.caltech.edu/~wolfe/normix.htm		
OC	hierárquica	sim	C
	http://www.barton.ebi.ac.uk/new/software.html		
SnoB	probabilística	sim	Fortran
	http://www.case.monash.edu.au/~dld/Snob.html		

Tabela 3.1: Alguns sistemas de *clustering*

3.8 Considerações Finais

Métodos de *clustering* são utilizados em diversas áreas de conhecimento como psicologia, biologia, sociologia, economia e engenharia, e recebem nomes distintos como análise Q, construção de tipologia, análise de classificação e taxonomia numérica.

Neste capítulo foram abordados tópicos relativos a *clustering* clássico, tais como técnicas, algoritmos e medidas de similaridade utilizadas. A abordagem de *clustering* baseada em similaridade tem produzido algoritmos eficientes os quais têm se mostrado úteis em muitas aplicações. Entretanto, a abordagem clássica tem algumas limitações significantes. Os resultados apresentados por essa abordagem são os clusters encontrados e algumas informações sobre similaridades numéricas entre exemplos que pertencem

a um mesmo cluster. Entretanto, esses algoritmos não fornecem descrição ou explicação "semântica" dos clusters descobertos.

Capítulo

4

Aprendizado Semi-Supervisionado

"Guerrear pela paz é como roubar pela honestidade"
Millôr Fernandes

Algoritmos capazes de aprender a partir de exemplos rotulados e não rotulados têm despertado ultimamente bastante interesse na comunidade científica. Essa nova forma de aprendizado, denominada aprendizado semi-supervisionado, é bastante útil quando apenas um pequeno número de exemplos rotulados encontra-se disponível, o que ocorre na grande maioria dos casos.

O aprendizado semi-supervisionado pode ser utilizado tanto em tarefas de classificação, quando os exemplos rotulados são utilizados no processo de rotulação de exemplos, quanto em tarefas de *clustering*, sendo os exemplos rotulados responsáveis por auxiliar o processo de formação de clusters.

Nesse capítulo é descrito o aprendizado semi-supervisionado e alguns dos algoritmos de aprendizado semi-supervisionado propostos na literatura.

4.1 Introdução

Como visto no Capítulo 2, caso existam exemplos rotulados, pode-se utilizar aprendizado supervisionado para induzir classificadores a partir desses exemplos. Caso contrário, quando os rótulos dos exemplos não são conhecidos, pode-se utilizar aprendizado não supervisionado com o objetivo de encontrar clusters (Capítulo 3). Já o aprendizado semi-supervisionado, uma recente área de pesquisa em AM, consiste em utilizar algoritmos que aprendem a partir de exemplos rotulados e não rotulados.

A grande motivação para esse tipo de aprendizado se dá pelo fato de exemplos não rotulados existirem em abundância e exemplos rotulados serem geralmente escassos. Além disso, a rotulação de exemplos pode ser custosa, como nos casos de indexação de vídeo, categorização de textos e diagnósticos médicos, entre outros. Uma outra motivação é que classificadores induzidos exclusivamente a partir de um pequeno conjunto de exemplos rotulados, geralmente, não apresentam boa precisão.

A idéia do aprendizado semi-supervisionado é então utilizar os exemplos rotulados para se obter informações sobre o problema e utilizá-las para guiar o processo de aprendizado a partir dos exemplos não rotulados (Bruce, 2001).

Aprendizado semi-supervisionado pode ser utilizado tanto em tarefas de classificação como em tarefas de *clustering* (Basu, Banerjee, & Mooney, 2002). Em uma classificação semi-supervisionada, a idéia é rotular, com uma certa margem de segurança, alguns dos exemplos no conjunto de exemplos não rotulados, os quais são posteriormente utilizados durante a fase de treinamento do classificador, freqüentemente resultando em uma classificação mais precisa (Blum & Mitchell, 1998). Já em *clustering* semi-supervisionado, os exemplos rotulados são utilizados no processo de formação dos clusters, servindo geralmente como um conhecimento de fundo, expresso, por exemplo, na forma de restrições, e resultando em melhores clusters.

Vários algoritmos de aprendizado semi-supervisionado têm sido propostos recentemente. A grande maioria deles tem como base algum algoritmo existente na literatura, o qual é modificado para tratar exemplos rotulados e não rotulados. Nas próximas seções são descritos sucintamente alguns desses algoritmos.

4.2 COP–*k-means*

Proposto por Wagstaff et al. (2001), o *COP–k-means* é uma variante do conhecido algoritmo *k-means*, descrito na Seção 3.5.1 na página 38. A diferença consiste na existência de um conhecimento de fundo inicial, descrito na forma de relações entre os exemplos, que é utilizado no processo de formação dos clusters.

Há dois tipos de relações:

- *Must-link*, que especifica que dois exemplos devem pertencer ao mesmo cluster;
- *Cannot-link*, que especifica que dois exemplos não devem pertencer ao mesmo cluster.

Essa abordagem é uma possível solução para o problema do aprendizado semi-supervisionado, pois permite a utilização tanto de exemplos rotulados, dos quais são extraídas as restrições, como de exemplos não rotulados.

Os clusters encontrados pelo *COP–k-means*, descrito pelo Algoritmo 1, devem respeitar todas as relações *must-link* e *cannot-link* impostas pelo usuário nos exemplos rotulados. Após a construção dos k clusters, cada exemplo do conjunto de exemplos não rotulados é associado ao cluster mais próximo.

Algoritmo 1 *COP–k-means*

Require: $E = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$: conjunto de exemplos com restrições;
k: número de clusters;
$\mathcal{R}$: conjunto de restrições;
procedure *COP–k-means* ( )
1: escolha os centróides iniciais dos clusters $C_1, C_2 \ldots C_k$;
2: **repeat**
3: **for all** $\mathbf{x}_i \in E$ **do**
4: associe $\mathbf{x}_i$ ao cluster C_j mais próximo, desde que as restrições em $\mathcal{R}$ não sejam violadas;
5: **end for**
6: atualize os centróides dos clusters que tiveram algum novo exemplo associado
7: **until** convergir;

4.3 SEEDED–*k-means*

Esse algoritmo, proposto por Basu et al. (2002), também é uma variante do *k-means*, e também particiona o conjunto de dados em k clusters. A diferença mais característica é o fato do *SEEDED–k-means* utilizar exemplos inicialmente rotulados como os centróides iniciais dos clusters, *i.e.*, as sementes (*SEED*, em inglês), e não escolhê-los aleatoriamente.

Dado um conjunto de exemplos E, toma-se um subconjunto $\mathcal{S} \subset E$ como sendo o conjunto de sementes. Na inicialização do algoritmo, o usuário é responsável por atribuir cada exemplo $\mathbf{x}_i \in \mathcal{S}$ a um dos k clusters a serem encontrados, dividindo o conjunto $\mathcal{S}$ em k subconjuntos $\mathcal{S}_l$, de tal forma que $\mathcal{S} = \cup_{l=1}^{k} \mathcal{S}_l$. Uma exigência do

algoritmo é que para cada cluster seja atribuído, no mínimo, uma semente. Dessa forma, cada cluster terá o seu centróide inicial inicializado como sendo a média das sementes a ele atribuídas pelo usuário. Os próximos passos do algoritmo são idênticos ao *k-means*.

4.4 CONSTRAINED–*k-means*

O algoritmo *CONSTRAINED–k-means*, também proposto por Basu et al. (2002), utiliza a mesma inicialização de centróides do algoritmo *SEEDED–k-means*, assemelhando-se muito com ele. A diferença consiste nos passos seguintes a inicialização dos centróides, nos quais os exemplos que fazem parte do conjunto de sementes, e que foram inicialmente associados a um dado cluster pelo usuário, não poderão ser associados a um outro cluster. Dessa forma, apenas os exemplos não selecionados como sementes terão o cluster correspondente reestimado.

Ao passo que no *SEEDED–k-means* as sementes podem vir a pertencer a clusters diferentes daqueles inicialmente associados, no *CONSTRAINED–k-means* isso não ocorre. Assim, o *CONSTRAINED–k-means* é mais adequado quando as sementes estão seguramente livres de ruídos. Caso contrário, recomenda-se o uso do *SEEDED–k-means*.

Os algoritmos anteriormente descritos são algoritmos de aprendizado semi-supervisionado que têm como objetivo melhorar o processo de *clustering*. A seguir, é descrita uma técnica de aprendizado semi-supervisionado que tem como objetivo melhorar o processo de classificação.

4.5 Co-Training

Co-Training é uma técnica que visa rotular exemplos automaticamente a partir de um pequeno conjunto de exemplos rotulados. Teoricamente simples, *Co-Training* se baseia na cooperação de dois algoritmos de aprendizado supervisionado. A idéia central consiste em um classificador rotular exemplos para o outro classificador, e vice-versa, incrementando a precisão de classificação dos exemplos.

Inicialmente *Co-Training* foi proposto para problemas nos quais os exemplos podem ser descritos por duas visões (Blum & Mitchell, 1998; Mitchell, 1999). Entretanto, a maioria dos problemas não possui tal característica. Para esses problemas, Goldman & Zhou (2000) propuseram um novo algoritmo de *Co-Training*. Ambas as estratégias são descritas a seguir.

4.5.1 *Exemplos Descritos por Duas Visões*

Sendo os exemplos descritos por duas visões V_1 e V_2, pode-se particionar o conjunto de atributos $\mathbf{X}$ em dois subconjuntos, $\mathbf{X}_{V1}$ e $\mathbf{X}_{V2}$ que descrevem os mesmos exemplos tal que $\mathbf{X} = \mathbf{X}_{V1} \cup \mathbf{X}_{V2}$ e $\mathbf{X}_{V1} \cap \mathbf{X}_{V2} = \emptyset$ como é ilustrado na Figura 4.1, na qual, por simplicidade, é considerado que $\mathbf{X}_{V1} = \{X_1, X_2, ..., X_i\}$ e $\mathbf{X}_{V2} = \{X_{i+1}, X_{i+2}, ..., X_m\}$.

	X_1	X_2	X_3	X_4	...	X_m	Y
E_1	x_{11}	x_{12}	x_{13}	x_{14}	⋮	x_{1m}	y_1
E_2	x_{21}	x_{22}	x_{23}	x_{24}	⋮	x_{2m}	?
E_3	x_{31}	x_{32}	x_{33}	x_{34}	⋮	x_{3m}	?
E_4	x_{41}	x_{42}	x_{43}	x_{44}	⋮	x_{4m}	y_4
E_5	x_{51}	x_{52}	x_{53}	x_{54}	⋮	x_{5m}	y_5
E_6	x_{61}	x_{62}	x_{63}	x_{64}	⋮	x_{6m}	?
⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮
E_n	x_{n1}	x_{n2}	x_{n3}	x_{n4}	⋮	x_{nm}	?

	X_1	X_2	...	X_i	Y
E_1	x_{11}	x_{12}	⋮	x_{1i}	y_1
E_2	x_{21}	x_{22}	⋮	x_{2i}	?
E_3	x_{31}	x_{32}	⋮	x_{3i}	?
E_4	x_{41}	x_{42}	⋮	x_{4i}	y_4
E_5	x_{51}	x_{52}	⋮	x_{5i}	y_5
E_6	x_{61}	x_{62}	⋮	x_{6i}	?
⋮	⋮	⋮	⋱	⋮	⋮
E_n	x_{n1}	x_{n2}	⋮	x_{ni}	?

	X_{i+1}	X_{i+2}	...	X_m	Y
E_1	$x_{1(i+1)}$	$x_{1(i+2)}$	⋮	x_{1m}	y_1
E_2	$x_{2(i+1)}$	$x_{2(i+2)}$	⋮	x_{2m}	?
E_3	$x_{3(i+1)}$	$x_{3(i+2)}$	⋮	x_{3m}	?
E_4	$x_{4(i+1)}$	$x_{4(i+2)}$	⋮	x_{4m}	y_4
E_5	$x_{5(i+1)}$	$x_{5(i+2)}$	⋮	x_{5m}	y_5
E_6	$x_{6(i+1)}$	$x_{6(i+2)}$	⋮	x_{6m}	?
⋮	⋮	⋮	⋱	⋮	⋮
E_n	$x_{n(i+1)}$	x_{n2}	⋮	x_{nm}	?

Figura 4.1: Duas descrições do conjunto de exemplos

Logo após, a idéia central é induzir independentemente duas hipóteses $\mathbf{h}_1$ e $\mathbf{h}_2$ utilizando, respectivamente, os exemplos rotulados descritos pelos atributos $\mathbf{X}_{V1}$ e $\mathbf{X}_{V2}$.

Após a fase de treinamento, cada hipótese $\mathbf{h}_1$ e $\mathbf{h}_2$ examinará o conjunto de exemplos não rotulados correspondente, com o intuito de encontrar exemplos que possam ser classificados com um alto grau de certeza. Caso encontre, o classificador rotulará

esses exemplos, incrementando o conjunto de exemplos rotulados. Ambas as hipóteses são então novamente induzidas a partir desse novo conjunto de exemplos rotulados, e o processo se repete quantas vezes for necessário. O processo é descrito pelo Algoritmo 2.

Algoritmo 2 *Co-training* para exemplos descritos por duas visões

Require: L_{V1} = conjunto de exemplos rotulados descritos pelos atributos $\mathbf{X}_{V1}$
L_{V2} = conjunto de exemplos rotulados descritos pelos atributos $\mathbf{X}_{V2}$
U_{V1} = conjunto de exemplos não rotulados descritos pelos atributos $\mathbf{X}_{V1}$
U_{V2} = conjunto de exemplos não rotulados descritos pelos atributos $\mathbf{X}_{V2}$

1: **loop**
2: $\mathbf{h}_1$ = hipótese induzida utilizando L_{V1}
3: $\mathbf{h}_2$ = hipótese induzida utilizando L_{V2}
4: Rotule, mas somente se houver um alto grau de certeza, exemplos de U_{V1} utilizando $\mathbf{h}_1$ e exemplos de U_{V2} utilizando $\mathbf{h}_2$
5: **if** não há novos exemplos rotulados **then**
6: **STOP**
7: **else**
8: Adicione todos esses novos exemplos rotulados a L_{V1} e L_{V2}
9: **end if**
10: **end loop**

Mas como esse algoritmo de *Co-Training* pode melhorar a precisão dos classificadores? A idéia é que quando $\mathbf{h}_1$, a qual foi induzida utilizando os exemplos em L_{V1}, consegue classificar um exemplo com um alto grau de certeza, esse exemplo rotulado por $\mathbf{h}_1$ é adicionado ao conjunto L_{V1}. O mesmo rótulo é utilizado para o exemplo correspondente e adicionado ao conjunto L_{V2}. Obviamente, o fato de $\mathbf{h}_1$ classificar um exemplo com certeza não implica que $\mathbf{h}_2$ também o fará. Nesse caso, $\mathbf{h}_1$ contribui com o aumento da precisão de $\mathbf{h}_2$ adicionando informação útil à base de treinamento. Similarmente, $\mathbf{h}_2$ poderá melhorar a base de treinamento, contribuindo então com a precisão de $\mathbf{h}_1$.

4.5.2 Rotulando Web Pages

Para ilustrar o uso de dados não rotulados no aprendizado supervisionado, Mitchell (1999) utiliza o algoritmo de *Co-Training* (Algoritmo 2) para rotular *web pages*. *Web pages* podem ser descritas utilizando atributos altamente redundantes. Como mostrado na Figura 4.2, *web pages* podem ser classificadas de acordo com as palavras em seu interior, ou de acordo com as palavras dos *hyperlinks* que apontam para a mesma, tais como "Computação" e "USP-São Carlos". Em muitos casos, as palavras presentes nos *hyperlinks* e as palavras presentes na *web page* são suficientemente redundantes para a

tarefa de classificação. Nesse caso, L_{V1} poderia conter como atributos palavras da *web page*, e L_{V2} palavras contidas nos *hyperlinks* que apontam para a mesma.

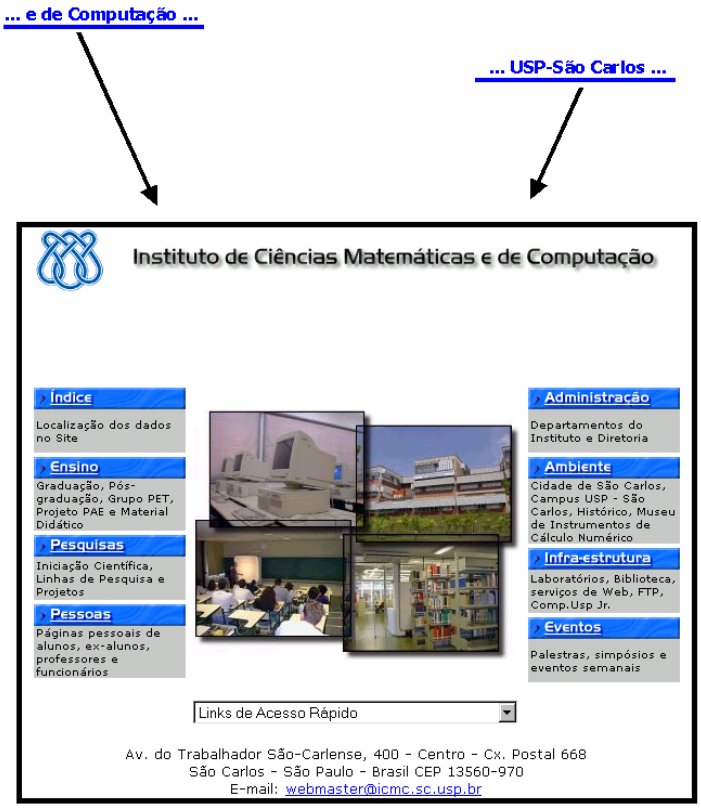


Figura 4.2: Duas descrições de uma *web page*

Blum & Mitchell (1998) mostram que a tarefa de classificação de *web pages* pode ter sua precisão melhorada com o algoritmo de *Co-Training* — Tabela 4.1. Nesse experimento a base de dados possui inicialmente apenas dezesseis exemplos rotulados e 800 *web pages* não rotuladas de *web sites* de departamentos de ciências da computação. Em cada iteração do algoritmo de *Co-Training*, é permitido a cada classificador adicionar um novo exemplo positivo e três negativos ao conjunto de exemplos rotulados. Após 30 (trinta) iterações, a precisão do classificador combinado atingiu 95.0%, contra 88.9% quando apenas 16 (dezesseis) exemplos rotulados foram utilizados.

	C_1	C_2	Classificador combinado
Treinamento supervisionado	12.9	12.4	11.1
Co-Training	6.2	11.6	5.0

Tabela 4.1: Taxa de erro para classificação de *web pages* (Blum & Mitchell, 1998)

4.5.3 Outras Aplicações

Uma vez comprovada a eficiência do *Co-Training* em classificação de *web pages*, é interessante determinar outras aplicações que podem fazer uso do mesmo método, tais como:

- Reconhecimento de fonemas na fala (de Sa & Ballard, 1997). Nesse caso, cada exemplo pode ser descrito por um sinal sonoro, L_{V1}, e por um sinal visual, L_{V2}, que é um vídeo mostrando os lábios do locutor. É importante salientar que tanto os atributos em L_{V1} quanto em L_{V2} devem ser capazes de descrever o fonema, o que caracteriza a redundância. Sendo assim, podem ser induzidos dois classificadores, $\mathbf{h}_1$ e $\mathbf{h}_2$, com o pequeno conjunto de dados rotulados, e permitir que cada um dos classificadores utilize os exemplos não rotulados para aumentar a precisão do outro.

- Reconhecimento de objetos em dados multimídia (Mitchell, 1999). Considerando a tarefa particular de identificar, em segmentos de vídeo, a presença de uma determinada pessoa, então, cada exemplo pode ser descrito por um frame contendo áudio, vídeo e texto em um particular intervalo. Sendo assim, L_{V1} pode ser o conjunto de atributos que descreve o áudio, L_{V2} o vídeo e L_{V3} o texto, caracterizando a redundância.

- Classificação de frases em classes semânticas (Riloff & Jones, 1999). Um exemplo seria classificar frases nominais como exemplos positivos ou negativos de localizações. Por exemplo, "São Paulo" seria um exemplo positivo, "amarelo" seria um negativo. Nesse caso, cada exemplo $\mathbf{x}$ seria uma frase do tipo "Nós estamos situados em Curitiba." A fatorização dos atributos que descrevem os exemplos em dois subconjuntos de atributos com características redundantes poderia ser feito de forma tal que L_{V1} refere-se ao substantivo da frase e L_{V2} ao contexto lingüístico no qual o exemplo é considerado. No caso do exemplo $\mathbf{x}$ considerado, o substantivo da frase seria "Curitiba" e o contexto "Nós estamos situados em".

4.5.4 Exemplos Descritos por Uma Única Visão

Como visto anteriormente, caso os exemplos possam ser descritos por duas ou mais visões, é possível utilizar exemplos não rotulados para aumentar a precisão do classificador. Entretanto, em muitas situações, cada exemplo possui uma única descrição, e o algoritmo apresentado na Seção 4.5.1 na página 53 não pode ser utilizado nesses casos.

Goldman & Zhou (2000) mostram como rotular exemplos quando estes são descritos por uma única visão. A princípio o processo é muito parecido com o proposto por Mitchell (1999), ou seja, dois classificadores que cooperam de forma que um melhore a precisão do outro. Entretanto, observando com detalhes os dois algoritmos, várias diferenças podem ser notadas.

A idéia desse novo algoritmo de *Co-Training* é utilizar dois algoritmos de aprendizado supervisionado, A e B, um conjunto U de exemplos não rotulados, um conjunto L de exemplos rotulados, um conjunto L_A, com os exemplos que B rotulou para A e um conjunto L_B, com os exemplos que A rotulou para B. Deve ser observado que somente serão rotulados exemplos após verificar diversas condições.

Cada iteração começa com o treinamento de A utilizando o conjunto de exemplos rotulados $L \cup L_A$ para se obter a hipótese $\mathbf{h}_A$. Analogamente, treina-se B com $L \cup L_B$ para se obter $\mathbf{h}_B$. Cada algoritmo escolherá, de sua respectiva hipótese, quais classes serão utilizadas para rotular exemplos em U para o outro algoritmo. Há dois testes que devem ser satisfeitos antes de se rotular algum dado. O primeiro deve garantir que a classe utilizada possua uma precisão no mínimo tão boa quanto a precisão das outras hipóteses. O segundo teste é para ajudar a prevenir a degradação da performance devido à possível presença de ruídos nos dados. Para todo exemplo em U que tenha uma classe equivalente em $\mathbf{h}_A$ e que passou por ambos os testes, A rotula o exemplo e coloca-o em L_B. B irá rotular exemplos da mesma forma que A. Intuitivamente, A deverá rotular um exemplo $\mathbf{x}$ em L_B somente se a confiança de A sobre o rótulo dado a $\mathbf{x}$ for melhor que a confiança de B em rotular o mesmo exemplo e a quantidade de exemplos rotulados for grande o suficiente para compensar um possível aumento de ruído na base de treinamento, decorrente de uma rotulação errada. Nesse momento chega ao fim uma iteração. O processo se repete até que L_A e L_B não sejam modificados durante uma iteração.

O próximo passo então é combinar $\mathbf{h}_A$ e $\mathbf{h}_B$ de forma a se obter uma hipótese final com uma melhor cobertura dos exemplos. Esse novo processo de *Co-Training* é descrito pelo Algoritmo 3 na próxima página.

4.5.5 *Exemplos Descritos por Duas Visões Aleatórias*

Uma alternativa para solucionar a dificuldade de ter duas visões diferentes para um mesmo conjunto de exemplos, é escolher aleatoriamente os atributos, de modo a construir dois conjuntos de atributos formando assim as duas partições necessárias para utilizar o método de *Co-Training*. Alguns resultados experimentais, utilizando essa técnica de *Co-Training*, são encontrados em Nigam & Ghani (2000). Na experiência

Algoritmo 3 *Co-Training* para exemplos descritos por uma única visão

Require: L = conjunto de exemplos rotulados
U = conjunto de exemplos não rotulados
$L_A = \emptyset$ //conjunto de exemplos rotulados pelo indutor B para serem utilizados pelo indutor A
$L_B = \emptyset$ //conjunto de exemplos rotulados pelo indutor A para serem utilizados pelo indutor B

1: **repeat**
2: $\mathbf{h}_A$ = hipótese induzida utilizando $L \cup L_A$
3: $\mathbf{h}_B$ = hipótese induzida utilizando $L \cup L_B$
4: L_{A_z} = exemplos ainda não rotulados de U, rotulados de acordo com certas condições, utilizando $\mathbf{h}_A$
5: L_{B_z} = exemplos ainda não rotulados de U, rotulados de acordo com certas condições, utilizando $\mathbf{h}_B$
6: $L_A = L_A \cup L_{B_z}$
7: $L_B = L_B \cup L_{A_z}$
8: **until** $L_{A_z} = \emptyset$ e $L_{B_z} = \emptyset$

descrita foi utilizada uma base de dados semi-artificial, constituída de 4 *newsgroups*. Na Tabela 4.2 é mostrado o número de exemplos rotulados e não rotulados apresentado inicialmente aos algoritmos.

Utilizando o algoritmo naive$\mathcal{B}ayes$ (Mitchell, 1997), foi induzido um classificador utilizando todos os exemplos rotulados (1006) e outro utilizando apenas 6 (seis) exemplos rotulados. No primeiro caso o erro do classificador foi de 3.9% enquanto que no segundo caso foi de 34.0%.

A seguir foi utilizado *Co-Training* com duas visões, as quais foram simuladas agrupando os 4 (quatro) *newsgroups* 2 (dois) a 2 (dois), formando assim as duas descrições necessárias. O algoritmo utilizou inicialmente 6 exemplos rotulados e 1000 não rotulados que foram aos poucos incrementando a base de exemplos rotulados. O erro obtido nesse caso foi de 3.7%.

Foi realizada uma experiência semelhante mas utilizando a abordagem descrita nessa seção na qual os atributos são selecionados aleatoriamente, para formar as duas visões dos exemplos, e foi obtido um erro de 5.5%.

Finalmente, utilizando EM (descrito na Seção 3.5.2 na página 40), foi obtido um erro de 8.9%. Nesse caso, os parâmetros iniciais foram estimados utilizando o algoritmo naive$\mathcal{B}ayes$ com os 6 exemplos rotulados. A cada iteração do EM , todos os exemplos são rotulados. Os exemplos que receberam rótulos muito confiáveis, são então adicionados ao conjunto de treinamento.

Assim, pode-se notar que mesmo com a seleção aleatória de atributos para a construção das duas descrições, *Co-Training* teve uma taxa de erro menor que naive$\mathcal{B}ayes$.

Algoritmo	Rotulados	Não rotulados	Erro
naive$\mathcal{B}ayes$	1006	0	3.9%
naive$\mathcal{B}ayes$	6	0	34%
Co-Training	6	1000	3.7%
Co-Training aleatório	6	1000	5.5%
EM	6	1000	8.9%

Tabela 4.2: Comparação de *Co-Training* aleatório com outros algoritmos

4.6 Outros Exemplos

Há ainda outros algoritmos de aprendizado semi-supervisionado que são variações do algoritmo EM , como o algoritmo apresentado por Bruce (2001), no qual os passos *expectation* e *maximization* são incrementados com uma variável denominada de pseudo-contador. Com essa alteração é possível estimar o rótulo dos dados não-rotulados. Entretanto, para que EM tenha um bom desempenho, os dados utilizados devem obedecer o modelo de "misturas finitas", o que restringe sua utilização para alguns poucos domínios.

Em Nigam et al. (2000) é apresentado uma extensão do algoritmo EM com naive$\mathcal{B}ayes$ que utiliza máxima probabilidade ou estimação máxima posterior de problemas com dados incompletos. No caso desse algoritmo, os dados não-rotulados são tratados como se fossem dados incompletos. Inicialmente o algoritmo naive$\mathcal{B}ayes$ induz o classificador apenas com os poucos 6 exemplos rotulados. Como EM também é um algoritmo *bayesiano*, no passo *expectation*, é inserido o classificador naive$\mathcal{B}ayes$, calculado anteriormente, para estimar algumas probabilidades, e no passo *maximization* os cálculos são realizados como no algoritmo EM básico. Os autores mostram alguns resultados experimentais obtidos utilizando textos de três situações diferentes do mundo real. Foi observado que a rotulação de exemplos utilizando esse método consegue reduzir o erro de classificação em até 30%.

Com algumas alterações no método de *Boosting*[1] (Schapire, 1990), é possível construir um algoritmo de aprendizado semi-supervisionado, como mostrado em d'Alché Buc et al. (2002). Um outro algoritmo, também baseado em *Boosting*, é o ASSEMBLE (Adaptative Semi-Supervise ensEMBLE) (Bennett, Demiriz, & Maclin, 2002). Usando árvores da decisão como classificador, esse algoritmo foi o primeiro de 34 algoritmos apresentados durante a competição de algoritmos semi-supervisionados NIPS (Neural

[1]Nesse método cada exemplo de treinamento possui um peso associado. A cada ciclo de iteração, uma hipótese é induzida a partir dos exemplos ponderados. Então, cada exemplo de treinamento tem seu peso associado modificado, considerando se ele foi ou não classificado corretamente pela hipótese induzida.

Information Processing Systems) 2001 (Kremer & Stacey, 2001).

Existem também alguns outros algoritmos que utilizam aprendizado por reforço e cortes em grafos (Blum & Chawla, 2001; Ivanov, Blumberg, & Pentland, 2001; Collins & Singer, 1999), e outros que utilizam SVM (*Support Vector Machines*) (Boser, Guyon, & Vapnik, 1992) com algumas alterações (Bennett & Demiriz, 1999; Demiriz & Bennett, 2000; Fung & Mangasarian, 1999; Seeger, 2002).

4.7 Considerações Finais

Neste capítulo foi apresentado o aprendizado semi-supervisionado. Nesse tipo de aprendizado, os algoritmos são capazes de aprender a partir de conjuntos de exemplos rotulados e não rotulados. A principal motivação para o seu uso se dá pelo fato de exemplos rotulados serem escassos, e caros. Viu-se também que os algoritmos de aprendizado semi-supervisionado têm dois objetivos, melhorar o processo de *clustering* ou melhorar o processo de classificação. No próximo capítulo é descrito um algoritmo, fruto do presente trabalho, que visa rotular exemplos a partir de um pequeno conjunto de exemplos rotulados, com o objetivo de melhorar o processo de classificação.

Capítulo

5

Algoritmo Proposto

"Se você pode sonhar com algo, então você pode fazê-lo."

Walt Disney

Dadas as limitações do aprendizado não supervisionado, como, por exemplo, a falta de uma descrição conceitual dos clusters encontrados, torna-se evidente a preferência pelo uso de algoritmos de aprendizado supervisionado. Entretanto, na grande maioria dos casos, o número de exemplos rotulados é muito pequeno, quando não inexistente, e sua geração pode ser de alto custo, inviabilizando o uso desse tipo de aprendizado.

Situações como essa incentivam o uso de algoritmos de aprendizado semi-supervisionado. Como descrito no capítulo anterior, esses algoritmos são capazes de aprender a partir de conjuntos de exemplos rotulados e não rotulados. Eles podem ser utilizados tanto em tarefas de classificação (aprendizado supervisionado), no processo de rotulação de exemplos, incrementando o conjunto de exemplos rotulados, quanto em tarefas de *clustering* (aprendizado não supervisionado), no processo de seleção inicial dos centróides ou de inserção de conhecimento de fundo, tendo como resultado a formação de

melhores clusters.

Neste trabalho é investigada a idéia de aprendizado semi-supervisionado para tarefas de classificação, considerando que exemplos que possuem a mesma classe apresentam uma tendência a se agrupar segundo alguma medida de similaridade. Com base nessa idéia, foi projetado e desenvolvido um algoritmo, denominado $k\text{-}means_{ki}$, a ser utilizado no processo de rotulação de exemplos a partir de um pequeno conjunto de exemplos rotulados para serem utilizados em tarefas de classificação. Neste capítulo é descrito o algoritmo proposto.

5.1 Introdução

Como mencionado, exemplos rotulados são escassos e de geração cara. Entretanto, pedir a um especialista que rotule um pequeno conjunto de exemplos muito representativos é uma tarefa aceitável. O algoritmo aqui proposto, a ser utilizado no processo de rotulação de exemplos, tem como requisito básico a existência de um pequeno conjunto de exemplos muito representativos rotulados por um especialista, com um alto grau de certeza.

Todo o processo de rotulação está baseado em uma idéia muito simples, composta praticamente de quatro etapas:

1. **Pré-Avaliação**: consiste na indução de um classificador, ou hipótese $\mathbf{h}_1$ inicial, a partir do pequeno conjunto de exemplos rotulados disponível e estimar sua precisão $ca(\mathbf{h}_1)$ utilizando 10–*fold cross–validation*, por exemplo.

2. ***Clustering***: consiste em encontrar clusters utilizando apenas os poucos exemplos rotulados, sendo que exemplos com classes diferentes não poderão ser agrupados em um mesmo cluster. Esse é um ponto muito importante, pois o rótulo dos exemplos não tem significado especial em aprendizado não supervisionado. Como visto na Seção 3.6 na página 44, clusters em aprendizado não supervisionado e classes em aprendizado supervisionado, não são, necessariamente, equivalentes. A idéia então é utilizar o atributo classe, tal que esses clusters agrupem apenas exemplos com o mesmo valor desse atributo.

3. **Rotulação**: consiste em analisar o conjunto de exemplos não rotulados com o intuito de determinar os exemplos que pertencem aos clusters encontrados, mas com um *alto grau de similaridade*. Uma vez escolhidos esses exemplos, eles serão rotulados com a classe dos exemplos do cluster correspondente, incrementando assim o conjunto de exemplos rotulados.

4. **Pós-Avaliação**: O novo conjunto de exemplos rotulados, *i.e.*, incrementado pelos exemplos rotulados no passo anterior, é fornecido ao mesmo indutor utilizado no passo 1 e uma nova hipótese $\mathbf{h}_2$ é induzida. Essa hipótese $\mathbf{h}_2$ terá então sua precisão $ca(\mathbf{h}_2)$ avaliada e, caso $ca(\mathbf{h}_2) > ca(\mathbf{h}_1)$ é admitido que o processo de rotulação obteve êxito, uma vez que a segunda hipótese teve sua precisão incrementada, e o processo pode ser repetido a partir do passo 2. Caso $ca(\mathbf{h}_1) > ca(\mathbf{h}_2)$, o processo de rotulação pára.

Como uma simples ilustração do processo proposto, considere a Figura 5.1 na página seguinte. Nessa figura os exemplos não rotulados são representados por •, e os exemplos rotulados são representados pela própria classe, cujos valores são, nesse caso, $+$ e $*$. Em (a) encontra-se representado o conjunto original de exemplos (rotulados e não rotulados) e em (b) são ilustrados os clusters encontrados utilizando somente os exemplos rotulados. Já em (c) os exemplos não rotulados estão, sempre que possível, alocados em seus respectivos clusters. Em (d) são escolhidos somente os exemplos "mais" similares, os quais receberão o rótulo do atributo classe dos exemplos do cluster correspondente (esses exemplos estão representados por •). O resultado final do processo, *i.e.*, o novo conjunto de exemplos rotulados é ilustrado em (e). Os novos exemplos rotulados estão destacados pelas setas.

Como pode ser observado na Figura 5.1, há a exigência de que os clusters encontrados a partir dos exemplos rotulados *não* agrupem exemplos com classes diferentes. Isso porque os exemplos a serem rotulados receberão a classe dos exemplos que formaram o cluster no qual ele foi inserido. Duas classes diferentes originando um só cluster inviabilizaria o uso do mesmo no processo de rotulação.

Uma vez definido como se dará o processo de rotulação, faz-se necessário escolher o algoritmo a ser utilizado no processo de *clustering*. A idéia inicial foi utilizar um algoritmo já existente na literatura. O primeiro algoritmo a ser cogitado foi o AUTOCLASS (Cheeseman & Stutz, 1990; Matsubara, Martins, & Monard, 2002). Entretanto, tal algoritmo não preencheu o requisito básico de garantir que exemplos com classes diferentes não sejam agrupados no mesmo cluster.

Uma alternativa para se atingir tal requisito é atribuir um peso muito alto ao atributo classe. Nesse caso, entretanto, corre-se o risco de que os clusters encontrados sejam baseados apenas nesse atributo, o que não é desejável.

Optou-se então por desenvolver um algoritmo "tipo *k-means*", *i.e.*, um algoritmo parecido com o *k-means* descrito na Seção 3.5.1 na página 38, porém, com algumas alterações necessárias para se atingir o objetivo proposto. Uma das alterações, e também a mais característica, é que os centróides iniciais são definidos pelos exemplos rotulados, e não mais definidos aleatoriamente. Dessa forma consegue-se garantir que os clusters

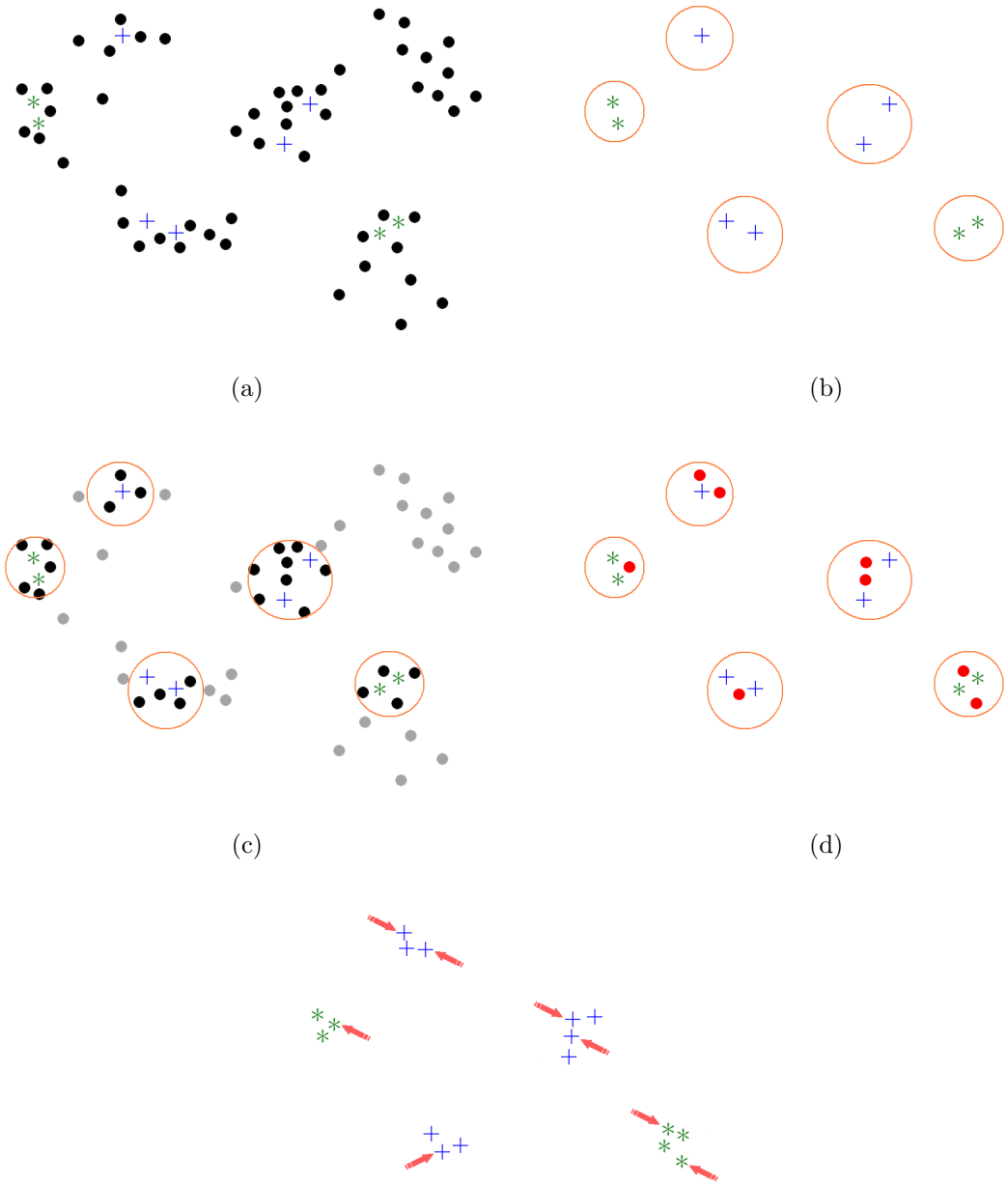


Figura 5.1: Processo de rotulação proposto: (a) conjunto original de exemplos (b) *clustering* (c) exemplos alocados em clusters (d) escolha dos exemplos mais similares (e) rotulação.

iniciais não possuirão exemplos com classes diferentes, uma vez que cada exemplo rotulado dará origem a um cluster. Esse novo algoritmo, denominado $k\text{-}means_{ki}$, é detalhado na próxima seção.

5.2 O Algoritmo k-$means_{ki}$

Baseado no *k-means*, o algoritmo k-$means_{ki}$, implementado na linguagem PERL (Wall, Christiansen, & Schwartz, 1996) (a mesma utilizada na implementação do Projeto DISCOVER) e fruto do presente trabalho, tem por objetivo rotular exemplos a partir de um pequeno conjunto de exemplos rotulados, com o intuito de melhorar o processo de classificação. A principal diferença entre os dois algoritmos é encontrada na fase de seleção de centróides. Ao passo que o *k-means* seleciona aleatoriamente seus centróides iniciais, o k-$means_{ki}$ define, como centróides iniciais, cada um dos exemplos rotulados disponíveis. Garante-se assim que exemplos rotulados com classes diferentes não serão agrupados em um mesmo cluster.

Uma outra diferença é quanto ao processo de *clustering* propriamente dito. Quando o *k-means* é utilizado, cada exemplo é associado ao cluster (centróide) mais próximo. No caso do k-$means_{ki}$, é estipulado *a priori* um *threshold* t. Esse *threshold* será o responsável pela associação exemplo/cluster. Ou seja, o exemplo somente poderá ser associado a um dado cluster caso esteja a uma distância menor ou igual a t de seu respectivo centróide.

Uma preocupação constante durante a fase de desenvolvimento do algoritmo, foi a de apenas rotular exemplos com um alto grau de certeza. Assim, ficou estipulado que caso um exemplo esteja a uma distância menor ou igual a t de dois ou mais centróides, ele apenas será rotulado caso os respectivos clusters forem formados por exemplos de mesma classe. Tal caso pode ser visto na Figura 5.2. Nessa figura os exemplos não rotulados, caracterizados por •, foram alocados em clusters originados por exemplos com classes diferentes. Logo, os mesmos não serão rotulados. Já os exemplos não rotulados caracterizados por •, serão rotulados com a classe do exemplo que originou o cluster no qual eles estiverem inseridos, uma vez que tais clusters foram originados por exemplos de mesma classe.

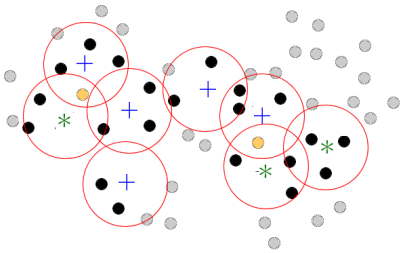


Figura 5.2: Exemplos pertencentes a dois ou mais clusters diferentes

O algoritmo k-$means_{ki}$ necessita, quando da sua execução, de três parâmetros, a serem definidos pelo usuário. O primeiro deles é a medida de similaridade a ser utilizada

no processo de *clustering*. O segundo parâmetro, já citado, é o valor de t, *i.e.*, o valor do *threshold* a ser utilizado no processo de rotulação dos exemplos. O último parâmetro diz respeito às diferentes versões do algoritmo k-$means_{ki}$. Tais parâmetros são detalhados nas seções seguintes.

5.2.1 Medida de Similaridade

Como mencionado na Seção 3.3 na página 30, há várias formas de se calcular a similaridade entre exemplos. No algoritmo k-$means_{ki}$, a medida de similaridade a ser utilizada na formação dos clusters e na definição do *threshold* t é definida pelo usuário. No presente trabalho foram implementadas as medidas de distância Euclideana e Chebychev. Como as medidas de similaridade foram implementadas em módulos independentes, a inclusão de novas métricas é uma tarefa fácil de ser realizada.

Na Figura 5.3 na próxima página são ilustradas as conseqüências do uso de tais medidas na formação dos clusters. Os pontos vermelhos ilustram a região delimitada pelos pontos distantes t do centróide de cada cluster de acordo com a medida de similaridade utilizada, *i.e.*, os clusters propriamente ditos. Em (a) foi utilizada a distância Euclideana, resultando em clusters circulares, e em (b) foi utilizada a distância Chebychev, resultando em clusters quadrados.

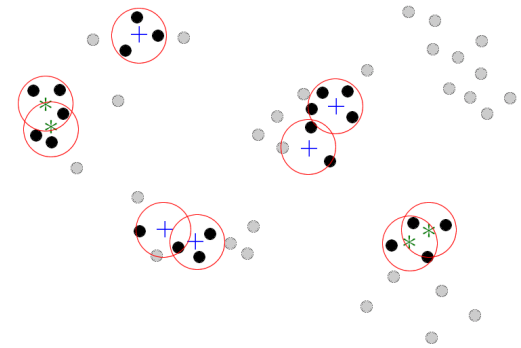
O algoritmo k-$means_{ki}$ foi concebido com o intuito de tratar exemplos cujos atributos possuem valores discretos, contínuos e/ou ausentes. Para tanto, todos os atributos contínuos são normalizados quando do início da execução do algoritmo. Assim, quando da comparação de dois atributos discretos $(x_i - y_i)$, será atribuído o valor 1 (um), que é a máxima diferença possível[1], caso eles possuam valores diferentes, ou 0 (zero), caso sejam iguais. No caso de um valor ausente, é automaticamente atribuído o valor 1 (um).

5.2.2 Threshold

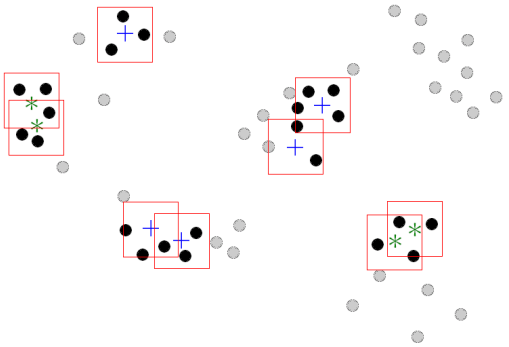
Um fator importante no algoritmo k-$means_{ki}$ é a definição do valor do *threshold*. Com certeza, definir um valor que represente um alto grau de similaridade não é uma tarefa simples. Essa definição torna-se ainda mais complicada quando da rotulação de exemplos a serem utilizados posteriormente por um algoritmo de aprendizado supervisionado, uma vez que rotular erroneamente um exemplo pode ter conseqüências indesejáveis.

No algoritmo k-$means_{ki}$, o *threshold* é estipulado pelo próprio usuário como um parâmetro de entrada. Assim, quanto maior for o valor de t, maior a probabilidade de

[1] Devido ao fato dos valores estarem normalizados



(a)



(b)

Figura 5.3: Conseqüências da utilização de diferentes medidas de similaridade

se rotular erroneamente um exemplo.

O usuário não estipula um valor absoluto para t, mas sim relativo. O valor absoluto do *threshold*, aqui denominado $t^{'}$ é calculado pelo $k\text{-}means_{ki}$ baseado em um vetor ordenado v contendo todas as distâncias entre todos os centróides (exemplos rotulados) e todos os exemplos não rotulados. Por exemplo, caso o usuário estabeleça o valor 5% para t, significa que serão passíveis de rotulação todos aqueles exemplos que estiverem a uma distância d de um ou mais centróides, se e somente se d estiver entre as 5% menores distâncias, *i.e.*, as 5% primeiras posições de v. Denominando o vetor formado pelas 5% primeiras posições do vetor ordenado v por $v^{'}$, o valor absoluto do *threshold*, denominado $t^{'}$, é dado pelo valor contido na última posição de $v^{'}$. Assim, serão então passíveis de rotulação todos aqueles exemplos que estiverem a uma distância d de algum centróide, tal que $d \leq t^{'}$. Para receber efetivamente um rótulo, basta que o exemplo não pertença simultaneamente a clusters formados por exemplos com classes diferentes.

Na Figura 5.4 é sumarizado o processo do cálculo do *threshold*. Uma vez calculadas, as distâncias são inseridas, ordenadamente, em um vetor (v). São selecionadas então

as $t\%$ primeiras distâncias de v ($v^{'}$), consideradas aceitáveis no processo de rotulação. O valor absoluto do *threshold*, denominado $t^{'}$, é dado pela distância contida na última posição de $v^{'}$, que está em destaque.

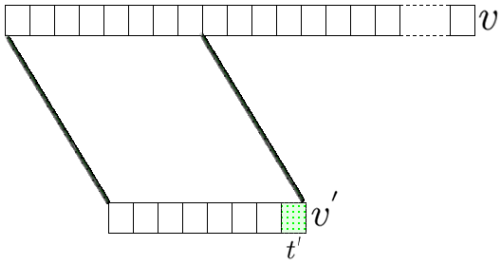


Figura 5.4: O cálculo do *threshold*

5.2.3 Versões

Adicionalmente, foram estudadas possíveis variações do algoritmo k-$means_{ki}$, o que resultou em quatro versões do mesmo. Tais versões possuem sutis diferenças, dentre as quais pode-se citar:

Versão 1: é a versão mais simples, na qual os centróides são calculados utilizando somente os exemplos iniciais rotulados pelo especialista. Nesse caso, o algoritmo k-$means_{ki}$ é executado somente uma vez, já que os novos exemplos rotulados não influenciam o processo de rotulação;

Versão 2: difere da primeira versão pelo fato de que os centróides são atualizados a cada iteração. Nessa versão os novos exemplos rotulados passam a influenciar no processo de rotulação. O perigo, nesse caso, é a existência de um rótulo errado, que pode acarretar em rótulos errados para exemplos a serem rotulados em cada iteração;

Versão 3: nessa versão há uma tentativa de se unir os centróides iniciais que estejam muito próximos, e que representam o mesmo conceito, *i.e.*, possuem a mesma classe (lembrando que os centróides são formados por exemplos rotulados). Dessa forma, dois centróides iniciais poderão resultar em um único cluster. Analogamente à versão 1, os centróides não são mais modificados, ou seja, não são atualizados quando da inclusão de novos exemplos não rotulados no cluster correspondente. Assim, o algoritmo k-$means_{ki}$ é executado uma única vez;

Versão 4: essa versão é praticamente uma união da versão 2 e versão 3, ou seja, busca-se unir os centróides iniciais muito próximos, e os mesmos são atualizados sempre que um novo exemplo não rotulado for inserido no cluster.

Entretanto, apenas as versões 1 e 2 produziram resultados aceitáveis. O problema encontrado com as versões 3 e 4 diz respeito à união dos centróides, o que acabava, nos experimentos realizados, acarretando em clusters grandes. Como conseqüência, a maioria dos exemplos não rotulados acabavam pertencendo simultaneamente a clusters com classes diferentes, não sendo, assim, rotulados. Esse caso é ilustrado na Figura 5.5.

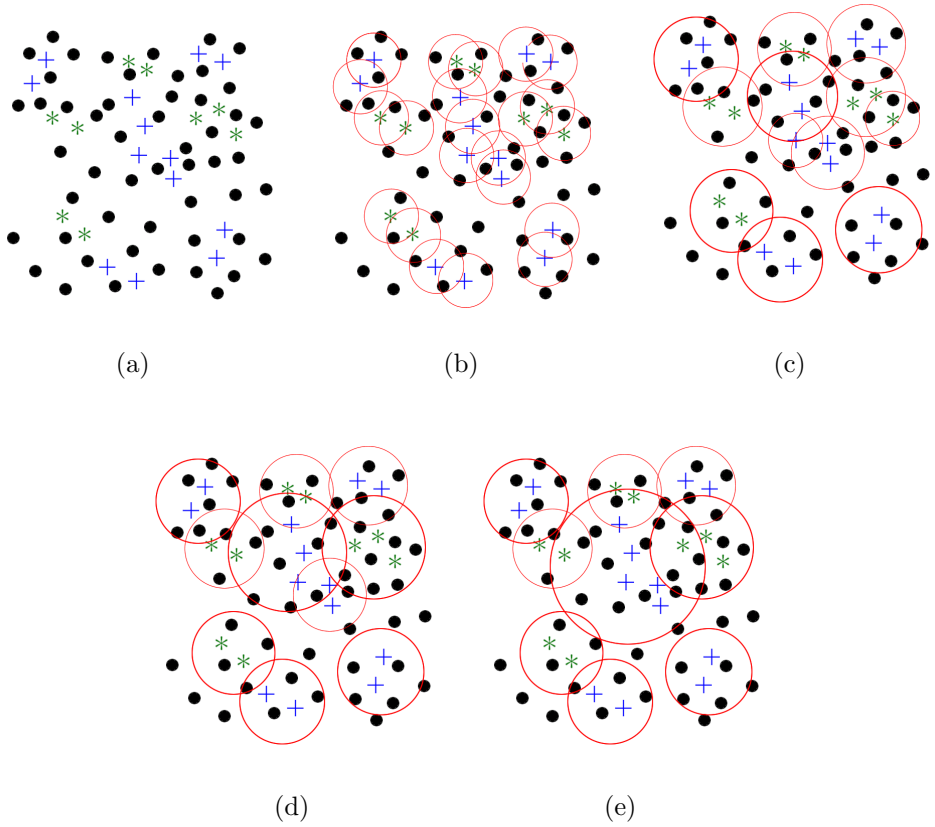


Figura 5.5: Conseqüências da união dos centróides (versões 3 e 4).

Em (a) é mostrado o conjunto de exemplos rotulados ($+$ e $*$) e não rotulados ($\bullet$). Em (b) podem ser notados os clusters correspondentes a cada exemplo rotulado. O processo de união dos centróides é ilustrado em (c), (d) e (e) — primeira, segunda e terceira iterações do algoritmo $k\text{-}means_{ki}$, respectivamente. Nesse exemplo pode-se verificar que o número de exemplos passíveis de rotulação caiu significantemente antes (34 exemplos) e depois (25 exemplos) da união dos centróides. Em alguns experimentos realizados com as quatro versões propostas, o número de exemplos rotulados após a fase de união dos centróides foi insignificante, quando comparado às versões que não possuem tal característica. Por esse fator, as versões 3 e 4 não são consideradas no presente trabalho. Essa situação — diminuição do número de exemplos rotulados — como será mostrado posteriormente, também é encontrada quando do aumento significativo do valor do

threshold, o que também acarreta em clusters maiores.

As versões 1 e 2 são descritas pelo Algoritmo 4.

Algoritmo 4 $k\text{-}means_{ki}$

Require: $E = \{(\mathbf{x}_i, y_i), i = 1, ..., N\}$: conjunto de exemplos rotulados;
$F = \{\mathbf{a}_i, i = 1, ..., M\}$: conjunto de exemplos não rotulados;
d: medida de distância a ser utilizada (Euclideana, Manhattan, etc);
t: *threshold* (distância mínima para rotular um exemplo);
v: versão do algoritmo;
$MD[N][M]$: matriz de distâncias entre os exemplos de E e F, onde $MD[i][j]$ especifica a distância entre $\mathbf{x}_i \in E$ e $\mathbf{a}_j \in F$;
procedure $k\text{-}means_{ki}$ (E, F, d, t, v, MD)

1: $R = \emptyset$: conjunto de exemplos rotulados pelo algoritmo;
2: escolha, como centróides iniciais dos clusters $C_1, C_2, \ldots, C_N$, os exemplos $\mathbf{x}_1$,$\mathbf{x}_2, \ldots$,$\mathbf{x}_N$, respectivamente;
3: **repeat**
4: **for all** $\mathbf{a}_i \in F$ **do**
5: $L_{clusters}$: lista ordenada com todos os clusters a uma distância d de $\mathbf{a}_i$, tal que $d \leq t$;
6: **if** todos os clusters em $L_{clusters}$ tiverem sido originados por exemplos rotulados com o mesmo rótulo r **then**
7: associe $\mathbf{a}_i$ ao cluster C_j mais próximo, tal que $C_j \in L_{clusters}$;
8: **end if**
9: **end for**
10: **if** $v = 2$ **then**
11: atualize os centróides dos clusters que tiveram algum novo exemplo associado;
12: atualize MD;
13: **end if**
14: **until** não ocorra nenhuma nova associação exemplo/cluster;
15: **for all** C_i **do**
16: **for all** $\mathbf{a}_j$ associado a C_i **do**
17: Rotule $\mathbf{a}_j$ com a classe do exemplo que originou o cluster C_i;
18: $R = R \cup \mathbf{a}_j$;
19: **end for**
20: **end for**
return R;

5.3 Implementação do Algoritmo

Para facilitar a integração com o projeto DISCOVER, o algoritmo $k\text{-}means_{ki}$ foi implementado em PERL (Wall, Christiansen, & Schwartz, 1996), como um módulo, o que facilita sua reutilização. Ao se integrar a tal projeto, a implementação foi facilitada pois foi possível fazer uso das diversas funcionalidades oferecidas pelo DISCOVER.

Como citado anteriormente, o projeto DISCOVER utiliza uma sintaxe padrão para representação dos dados. Essa sintaxe, denominada DSX — DISCOVER DATASET SINTAX — utiliza arquivos texto para declarar os atributos e seus respectivos tipos, e os valores que esses atributos assumem em um conjunto de dados (Batista, 2002). Os atributos são declarados em um arquivo com extensão `.names`. Os valores que esses atributos assumem em um conjunto de dados são declarados em um outro arquivo com a extensão `.data`.Os dois arquivos devem possuir o mesmo nome, se diferenciando apenas pela extensão. Conseqüentemente, o algoritmo *k-means*$_{ki}$ faz uso da sintaxe DSX, e seus arquivos de saída respeitam tal formato.

Estando os dados na sintaxe DSX, pode-se utilizar o módulo Core, pertencente à DOL — DISCOVER OBJECT LIBRARY — para carregar e pré-processar os dados (Batista & Monard, 2003b). O módulo Core é o maior módulo da biblioteca DOL, e é responsável por carregar um conjunto de dados em uma estrutura e prover uma gama de métodos capazes de realizar dezenas de operações sobre essa estrutura. Posteriormente, os dados na estrutura podem ser armazenados em mais de uma dezena de sintaxes diferentes utilizadas por alguns dos principais indutores existentes. Ao utilizar a biblioteca DOL, não foi necessário se ater às questões referentes ao carregamento, e pré-processamento dos dados, durante a implementação do algoritmo *k-means*$_{ki}$.

Após o carregamento dos dados, o processo é controlado pelo módulo *k-means*$_{ki}$, responsável pela rotulação dos exemplos. Esse módulo requisita a entrada de dois arquivos, um deles contendo apenas exemplos rotulados — `rotulados.data` — e o outro contendo exemplos não rotulados — `nao_rotulados.data` — cada qual com o seu respectivo arquivo `.names`.

Encerrado o processo de rotulação, o algoritmo *k-means*$_{ki}$ tem como saída 4 (quatro) arquivos texto, descritos a seguir.

- `k_rotulados.data`:contém todos os exemplos previamente rotulados que encontram-se no arquivo `rotulados.data`, mais os exemplos rotulados pelo *k-means*$_{ki}$;
- `k_rotulados.names`:contém a descrição de todos os atributos contidos no arquivo `k_rotulados.data`;
- `k_nao_rotulados.data`:contém todos os exemplos previamente não rotulados no arquivo `nao_rotulados.data`, que não foram rotulados pelo *k-means*$_{ki}$;
- `k_nao_rotulados.names`:contem a descrição de todos os atributos contidos no arquivo `k_nao_rotulados.data`.

O próximo passo consiste da análise dos rótulos atribuídos pelo algoritmo *k-means*$_{ki}$.

Novamente a implementação fez uso das funcionalidades do projeto DISCOVER, mais precisamente do ambiente computacional SNIFFER (Batista & Monard, 2003a).

O ambiente computacional SNIFFER automatiza a avaliação experimental, e é totalmente integrado com diversos indutores — Tabela 5.1. Com isso, é possível comparar o desempenho do novo conjunto de exemplos rotulados pelo $k\text{-}means_{ki}$— `k_rotulados.data`— aplicado a diversos indutores.

Indutores
$\mathcal{C}$4.5 (Quinlan, 1993)
$\mathcal{C}$4.5rules (Quinlan, 1987)
$\mathcal{ID}$3 (Quinlan, 1986)
$\mathcal{CN}$2 (Clark & Boswell, 1991)
new$\mathcal{ID}$ (Boswell, 1990)

Tabela 5.1: Indutores suportados pelo DISCOVER

O SNIFFER também se utiliza da DOL para a etapa de carregamento e pré-processamento dos dados. Nesse caso, um dos pré-processamentos aplicados sobre os dados consiste na transformação dos dados na sintaxe DSX para a sintaxe exigida pelo indutor a ser utilizado. Feito isso, o SNIFFER executa o(s) indutor(es) especificado(s) pelo usuário, realiza validação cruzada, coleta a(s) sua(s) taxa(s) de erro, e constrói um relatório com todos os resultados.

Os resultados desse relatório do SNIFFER são utilizados pelo $k\text{-}means_{ki}$ para calcular a precisão dos classificadores. Caso o resultado mostre que o novo conjunto de exemplos rotulados — `k_rotulados.data` — foi capaz de aumentar a precisão do classificador quando comparado àquele previamente induzido com o conjunto de exemplos inicialmente rotulados — `rotulados.data` — os arquivos de saída do algoritmo tornam-se arquivos de entrada, e o processo se repete.

Na Figura 5.6 é apresentado, de forma resumida, todo o processo de rotulação proposto. Nela, o fluxo azul indica a indução do primeiro classificador a partir do conjunto de exemplos inicialmente rotulados e sua avaliação. O fluxo vermelho indica a execução do $k\text{-}means_{ki}$ a partir dos conjuntos de exemplos fornecidos (rotulados e não rotulados), e a posterior indução, e avaliação, de classificadores a partir da saída do algoritmo. O fluxo verde indica a continuação da execução do $k\text{-}means_{ki}$ sempre que possível, *i.e.*, sempre que os exemplos rotulados pelo algoritmo forem capazes de aumentar a precisão do classificador. Deve ser salientado que nesse caso, os arquivos de saída do $k\text{-}means_{ki}$ são fornecidos ao algoritmo como arquivos de entrada, e o fluxo vermelho é repetido.

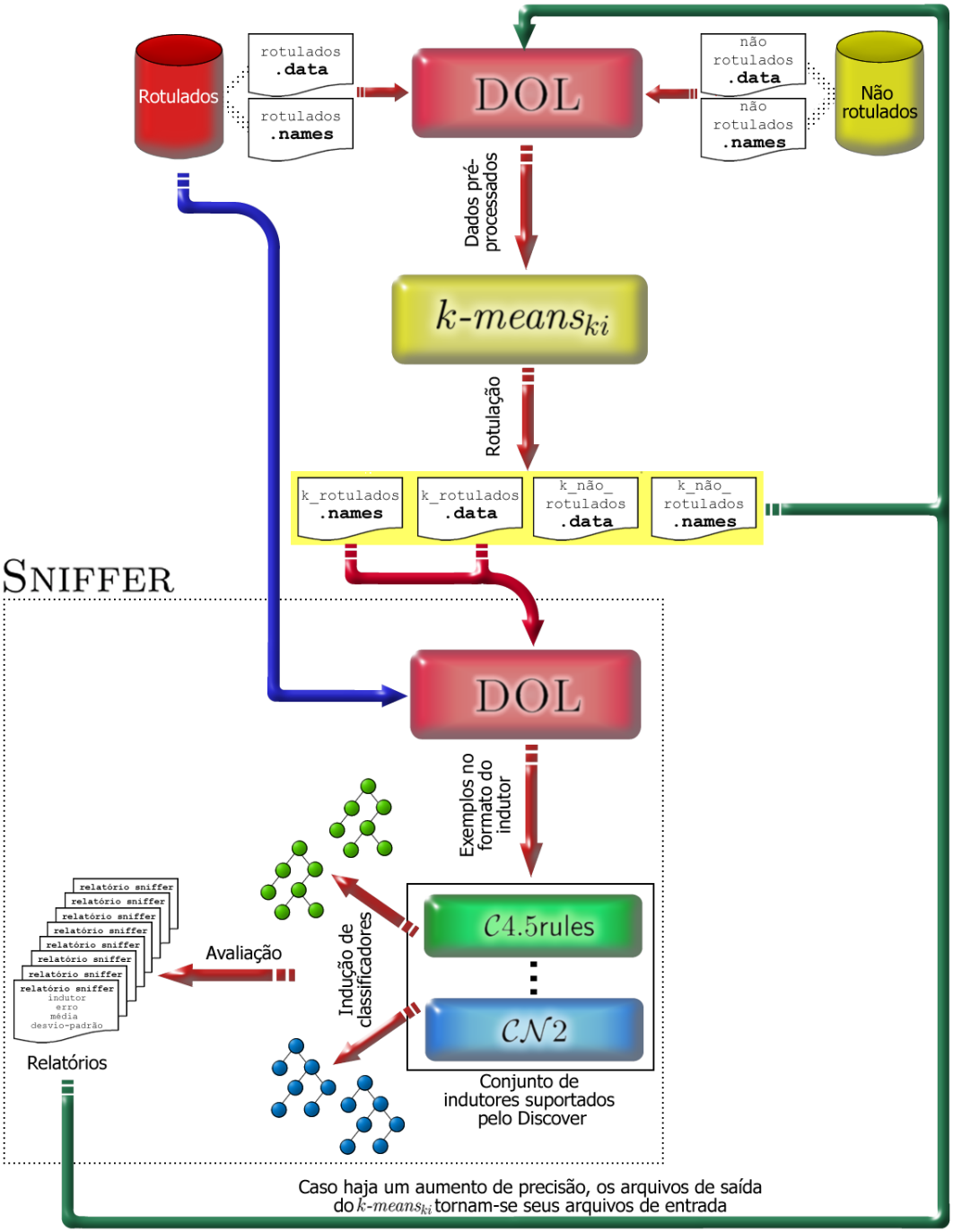
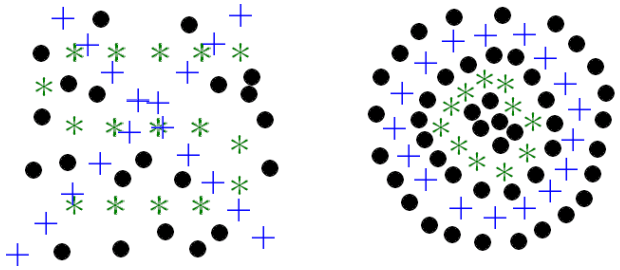


Figura 5.6: Descrição do processo de rotulação proposto

5.4 Bias *do Algoritmo*

O algoritmo k-$means_{ki}$, independentemente da versão considerada, possui um *bias* muito característico. Ele considera que exemplos que expressam o mesmo conceito, *i.e.*, possuem a mesma classe, estão próximos, segundo a medida de similaridade utilizada na sua execução.

Sendo assim, o seu uso não é aconselhável em conjuntos de dados que não possuem tal característica. Na Figura 5.7 é ilustrado esse aspecto. Nessa figura estão representados dois possíveis conjuntos de dados ((a) e (b)), contendo exemplos rotulados ($+$ e $*$) e não rotulados ($\bullet$). Em ambos os casos, exemplos que possuem a mesma classe não estão necessariamente próximos, e o que é pior, algumas vezes estão mais próximos de exemplos de outras classes do que da sua própria classe. Nesse caso a performance do k-$means_{ki}$ para auxiliar na rotulação de exemplos será ruim, e seu uso não é recomendado.



(a) (b)

Figura 5.7: Situação inadequada para uso do algoritmo k-$means_{ki}$

5.5 Considerações Finais

Neste capítulo foi apresentado o algoritmo k-$means_{ki}$. Tal algoritmo busca rotular exemplos automaticamente a partir de um pequeno conjunto de exemplos rotulados. A idéia é utilizá-lo para incrementar um conjunto de exemplos rotulados, viabilizando assim o uso de algoritmos de aprendizado supervisionado. Entretanto, seu uso é limitado aos casos nos quais os exemplos com a mesma classe estão próximos segundo a métrica utilizada pelo algoritmo.

A avaliação experimental do algoritmo é apresentada no próximo capítulo.

Capítulo

6

Avaliação Experimental

"A grandeza não consiste em receber honras,
mas em merecê-las."
Aristóteles

Neste capítulo são descritos os experimentos realizados para verificar o comportamento do algoritmo $k\text{-}means_{ki}$. Todos os experimentos foram realizados com conjuntos de exemplos rotulados. Desses conjuntos de exemplos são extraídos alguns exemplos rotulados e os restantes têm seu rótulo extraído e, durante a simulação, são assumidos como exemplos não rotulados. Com isso, é possível comparar o rótulo atribuído a um dado exemplo pelo algoritmo $k\text{-}means_{ki}$, com o seu rótulo original.

A seguir são descritos os conjuntos de dados utilizados nas experiências, como foram organizados os experimentos, bem como os resultados obtidos.

6.1 Conjuntos de Dados Utilizados

Os experimentos foram realizados utilizando vários conjuntos de dados naturais de diferentes domínios. Os conjuntos de dados foram obtidos do repositório da UCI (Merz & Murphy, 1998).

Com o objetivo de auxiliar nas comparações, os conjuntos de dados escolhidos possuem diferentes tipos de atributos. Os atributos podem ser contínuos ou nominais, o que não implica que esses tipos apareçam isoladamente em cada conjunto de dados. Foram ainda selecionados conjuntos de dados que contém valores ausentes para validar o tratamento desse tipo de problema pelo algoritmo $k\text{-}means_{ki}$. Segue uma descrição geral de cada conjunto de dados utilizado nos experimentos realizados.

Breast–Cancer 2 Esse conjunto de dados é um dos conjuntos nomeados *Breast Cancer* que estão na UCI, no qual o problema é predizer sobre a recorrência ou não recorrência de câncer de mama.

Hepatitis Esse conjunto de dados está relacionado à predição da expectativa de vida de pacientes com hepatite.

Breast–Cancer Nesse conjunto de dados o problema é predizer quando uma amostra de tecido da mama extraído de uma paciente possui tumor benigno ou maligno.

Na Tabela 6.1 é apresentado um resumo das principais características de cada um dos conjuntos de dados. É mostrado, o número de exemplos (#Exemplos), número e percentual de exemplos duplicados (aparecem mais que uma vez) ou conflitantes (possuem o mesmo conjunto atributo-valor mas diferente classe), número de atributos (#Atributos) contínuos e nominais, o erro majoritário e se o conjunto de dados tem ao menos um valor ausente[1]. Os conjuntos de dados são apresentados em ordem crescente do número de atributos. Na Figura 6.1 é mostrada a dimensão dos conjuntos de dados, *i.e.* o número de atributos e o número de exemplos de cada um deles. Deve ser observado que devido a grande variação na dimensão, o número de exemplos na Figura 6.1 é apresentado na escala logarítmica, *i.e.*, $\log_{10}$(#Exemplos).

6.2 Organização dos Experimentos

Os experimentos foram realizados com dois objetivos:

1. analisar a validade dos rótulos encontrados pelo algoritmo $k\text{-}means_{ki}$;

[1] Essas informações foram obtidas utilizando o utilitário $\mathcal{MLC}$++*info*.

Conjuntos de dados	#Exemplos	Duplicados ou conflitantes (%)	#Atributos (cont., nom.)	Classe	Erro Majoritário	Valores Ausentes
Breast–Cancer 2	285	2 (0.7%)	9 (4,5)	recurrence (29.47%) no-recurrence (70.53%)	29.47% no-recurrence	Sim
Hepatitis	155	0 (0%)	19 (6,13)	die (20.65%) live (79.35%)	20.65% live	Sim
Breast–Cancer	699	8 (1.15%)	9 (9,0)	2 (65.52%) 4 (34.48%)	34.48% 2	Sim

Tabela 6.1: Características dos conjuntos de dados

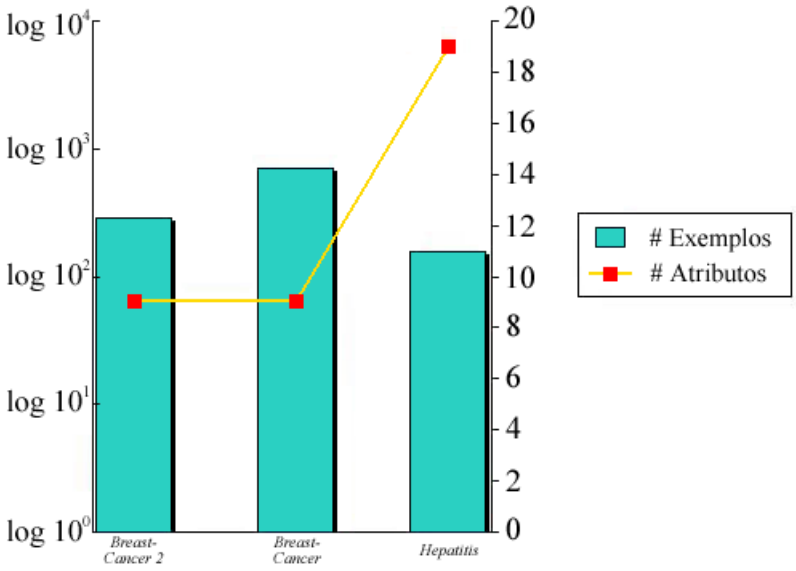


Figura 6.1: Dimensão dos conjuntos de dados

2. analisar a utilidade dos novos exemplos rotulados na indução de um classificador.

Para se atingir o primeiro objetivo, cada um dos conjuntos de dados utilizados nos experimentos é particionado em dois subconjuntos. Um deles é grande, e seus exemplos têm a classe extraída, já o outro é pequeno, e seus exemplos mantêm o rótulo. Dessa forma, simula-se a situação para a qual o k-$means_{ki}$ foi concebido, ou seja, aquela em que está disponível uma grande quantidade de exemplos não rotulados, e um pequeno conjunto de exemplos rotulados. Para esse primeiro passo — particionar o conjunto de exemplos — foi implementado um módulo denominado splitDB.

Após submeter esses dois subconjuntos ao k-$means_{ki}$, a saída do algoritmo, *i.e.*, o novo conjunto de exemplos rotulados, é analisada. Essa análise é uma tarefa simples, e consiste da comparação do rótulo atribuído pelo algoritmo k-$means_{ki}$ a um dado exemplo com o seu rótulo original conhecido. Dessa forma, pode-se medir a taxa de acerto do algoritmo k-$means_{ki}$. Para essa tarefa de análise foi implementado um módulo denominado analysis.

Uma vez que um dos objetivos deste trabalho é melhorar a tarefa de classificação, o segundo objetivo consiste na análise da utilidade desse novo conjunto de exemplos

rotulados quando da indução de um classificador. Para tanto, são induzidos dois classificadores. O primeiro deles é induzido a partir do pequeno subconjunto de exemplos rotulados, inicialmente utilizado pelo algoritmo k-$means_{ki}$. O segundo classificador é induzido a partir desse mesmo conjunto de exemplos rotulados, adicionado do novo conjunto de exemplos rotulados pelo k-$means_{ki}$. Dessa forma, pode-se comparar se o novo conjunto de exemplos rotulados foi capaz de melhorar a tarefa de classificação.

Deve ser salientado que a segunda parte dos experimentos — análise da utilidade dos novos exemplos rotulados quando da indução de um classificador — não exprime o que realmente ocorre em um processo não simulado do uso do k-$means_{ki}$.

O algoritmo k-$means_{ki}$, quando utilizado em situações reais, terá sua execução finalizada quando os exemplos por ele rotulados não forem mais capazes de aumentar a precisão do classificador. Para tanto, inicialmente é induzido um classificador $\mathbf{h}$ a partir do pequeno conjunto de exemplos rotulados disponível, e sua precisão $ca(\mathbf{h})$ é estimada utilizando 10–*fold cross–validation* ou *leave–one–out*, por exemplo. O próximo passo então consiste da execução do k-$means_{ki}$, o que, se possível, resultará em um novo conjunto de exemplos rotulados. Toma-se então esse novo conjunto de exemplos rotulados, adicionado dos exemplos inicialmente rotulados, e induz-se um novo classificador $\mathbf{h}'$. Esse novo classificador $\mathbf{h}'$ terá sua precisão $ca(\mathbf{h}')$ estimada e, caso $ca(\mathbf{h}') > ca(\mathbf{h})$ é admitido que o processo de rotulação obteve êxito. Nesse caso, todos os exemplos rotulados agora disponíveis são novamente fornecidos ao k-$means_{ki}$, e todo o processo se repete até que $ca(\mathbf{h}') < ca(\mathbf{h})$.

Entretanto, nos experimentos realizados neste trabalho, os classificadores têm sua precisão estimada utilizando-se, como conjunto de teste, todos aqueles exemplos que não foram rotulados pelo k-$means_{ki}$ [2]. Como mencionado anteriormente, a precisão assim estimada nas simulações realizadas, não é a precisão que seria calculada em uma aplicação real, já que, em situações reais, é desconhecido o rótulo dos exemplos não rotulados pelo k-$means_{ki}$. Assim, em situações reais, a precisão $ca(\mathbf{h}')$ somente pode ser estimada utilizando os exemplos rotulados pelo algoritmo.

Resumidamente, as simulações podem ser decompostas em seis passos, descritos a seguir, e ilustrados na Figura 6.2 na próxima página.

1. Particionamento do conjunto de exemplos: consiste em particionar o conjunto de exemplos original em dois subconjuntos de exemplos (rotulados e não rotulados);

2. Primeira indução do classificador: consiste na indução de um classificador a partir do conjunto de exemplos rotulados, obtido no passo anterior;

[2] Na realidade, foram feitas algumas variações de uso do conjunto de teste, as quais serão explicadas nos experimentos correspondentes.

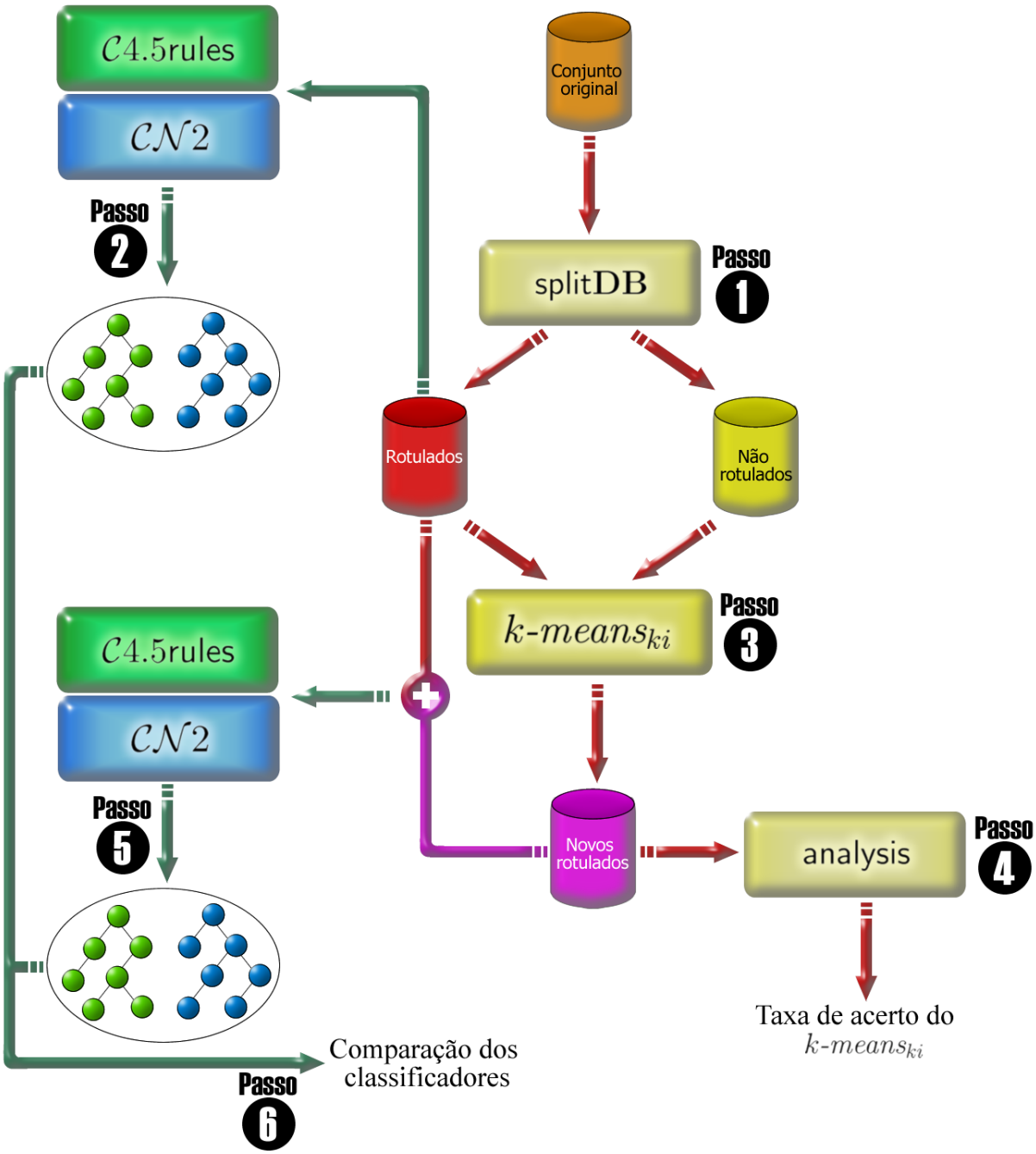


Figura 6.2: Descrição dos experimentos

3. Execução do k-$means_{ki}$: consiste na execução do algoritmo k-$means_{ki}$ com os dois subconjuntos de exemplos obtidos no passo 1;

4. Análise dos rótulos: consiste em analisar os rótulos atribuídos pelo algoritmo k-$means_{ki}$, estimando a taxa de acerto;

5. Segunda indução do classificador: consiste na indução de um classificador a partir do conjunto de exemplos rotulados obtido no passo 1 adicionado do conjunto de exemplos rotulados obtido no passo 3;

6. Comparação: consiste na comparação do classificador obtido no passo 2 com o classificador obtido no passo 5, tentando determinar qual tem melhor desempenho.

Os indutores utilizados nos passos 2 e 5 foram $\mathcal{C}$4.5rules (Quinlan, 1993) e $\mathcal{CN}2$ (Clark & Niblett, 1987; Clark & Boswell, 1989, 1991), descritos brevemente a seguir.

$\mathcal{C}$4.5rules: o $\mathcal{C}$4.5rules parte da árvore de decisão originalmente produzida pelo algoritmo $\mathcal{C}$4.5 (Quinlan, 1993) e deriva dessa árvore um conjunto de regras não ordenadas na forma **if** <*complexo*> **then** <*classe*>.

É importante notar que o $\mathcal{C}$4.5rules generaliza o conhecimento representado na árvore de decisão removendo condições descartáveis — *i.e.*, condições irrelevantes que não afetam a conclusão — sem afetar a precisão e retém as melhores regras.

$\mathcal{CN}2$: o $\mathcal{CN}2$ é um algoritmo de AM que induz um conjunto de regras não ordenadas no mesmo formato do $\mathcal{C}$4.5rules (Baranauskas & Monard, 2000). Para classificar um novo exemplo utilizando as regras induzidas, todas as regras são testadas e o conjunto das regras satisfeitas é obtido. Se mais de uma classe for predita pelas regras satisfeitas, então é considerado o número de exemplos cobertos por cada uma dessas regras. Depois somam-se esses valores para encontrar a classe mais provável.

Na Tabela 6.2 é apresentado o erro e o desvio padrão, calculado utilizando 10–*fold cross–validation*, dos classificadores induzidos por $\mathcal{C}$4.5rules e $\mathcal{CN}2$ a partir dos conjuntos de dados descritos na Tabela 6.1 na página 77. Ou seja, o erro dos classificadores induzidos utilizando todo o conjunto de exemplos rotulados.

	Indutores	
Conjuntos de dados	$\mathcal{C}$4.5rules	$\mathcal{CN}2$
Breast–Cancer 2	27.7±1.7	27.0±2.3
Hepatitis	20.5±3.0	16.2±1.8
Breast–Cancer	4.3±0.6	4.9±0.8

Tabela 6.2: Erro dos classificadores induzidos por $\mathcal{C}$4.5rules e $\mathcal{CN}2$ utilizando 10–*fold cross–validation* com todo o conjunto de exemplos

Como ilustrado na Figura 6.2 na página precedente, $\mathcal{C}$4.5rules e $\mathcal{CN}2$ são utilizados no passo 2 para induzir os respectivos classficadores utilizando somente o subconjunto

inicial de exemplos rotulados, bem como no passo 5, utilizando esse subconjunto inicial adicionado do novo subconjunto de exemplos rotulados pelo algoritmo $k\text{-}means_{ki}$.

Uma observação a ser feita sobre as experiências realizadas diz respeito à maneira como o conjunto de exemplos original é particionado em exemplos rotulados e não rotulados. Neste trabalho, os conjuntos de dados foram particionados de duas formas diferentes. A primeira delas consiste da seleção aleatória dos exemplos que constituem o subconjunto de exemplos rotulados. Na segunda, o conjunto de exemplos original é inicialmente processado utilizando o algoritmo AUTOCLASS para encontrar clusters nesse conjunto. Por ser um algoritmo de *clustering* que utiliza a técnica probabilística, ao fim da execução o AUTOCLASS associa a cada exemplo a probabilidade deste pertencer a cada um dos clusters encontrados. Dentre esses clusters, foram considerados apenas aqueles que agrupam exemplos que têm o mesmo rótulo. Após, foram escolhidos alguns dos exemplos rotulados nesses últimos clusters que têm probabilidade 1 (um), *i.e.*, 100%, de pertencer a um desses clusters.

Independentemente do tipo de particionamento adotado, no subconjunto de exemplos rotulados escolhidos sempre foi mantida a distribuição de classes encontrada no conjunto de exemplos original. Vale lembrar que o subconjunto de exemplos não rotulados é composto por todos os exemplos, cujo rótulo é extraído, que não foram selecionados para participar do subconjunto inicial de exemplos rotulados.

Um outro aspecto a ser ressaltado, é que foram considerados todos os atributos dos exemplos, sem aplicar previamente algum processo para extrair os atributos mais relevantes dos dados.

Todos os experimentos foram repetidos 5 (cinco) vezes, sendo que em cada vez foi utilizado um subconjunto diferente de exemplos rotulados. Foram também utilizados *thresholds* diferentes (5%, 10% e 20%) bem como uma proporção diferente de exemplos rotulados. A seguir são mostrados os resultados obtidos realizando seleção aleatória de exemplos rotulados.

6.3 Resultados Obtidos com Seleção Aleatória de Exemplos Rotulados

Nas tabelas a seguir são mostrados os resultados obtidos para cada conjunto de dados, onde:

Distância: indica a medida de distância utilizada — Euclideana ou Chebychev. Deve ser observado que para os conjuntos de dados *Breast–Cancer 2* e *Hepatitis*, os

quais possuem atributos discretos, nenhum novo exemplo foi rotulado utilizando a distância Chebychev;

***Threshold*:** indica o *threshold* utilizado no experimento — 5%, 10% e 20%;

Versão: a versão do algoritmo *k-means*$_{ki}$ utilizada — v1 indica a versão 1 e v2 a versão 2;

Total: mostra a média e o desvio padrão do número de exemplos rotulados nos 5 (cinco) experimentos realizados;

% Rotulados: mostra a percentagem aproximada de novos exemplos rotulados, considerando o número total de exemplos não rotulados desse conjunto de dados para esse experimento específico. Essa percentagem é calculada considerando o valor médio de exemplos rotulados pelo algoritmo;

Errados: mostra a média e o desvio padrão do número de exemplos rotulados erroneamente, considerando o número de exemplos rotulados (**Total**) pelo algoritmo nos 5 (cinco) experimentos realizados;

% Errados: mostra a percentagem aproximada de exemplos rotulados erroneamente, considerando o número médio de exemplos rotulados pelo algoritmo (**Total**).

Pós-Rotulação: mostra o erro e o desvio padrão dos classificadores induzidos por $\mathcal{C}4.5$rules e $\mathcal{CN}2$ respectivamente, utilizando como conjunto de treinamento o subconjunto inicial de exemplos rotulados mais o subconjunto de exemplos rotulados pelo *k-means*$_{ki}$. Como conjunto de teste é utilizado o subconjunto de exemplos que não foram rotulados.

Pré-Rotulação: mostra o erro e o desvio padrão dos classificadores induzidos respectivamente por $\mathcal{C}4.5$rules e $\mathcal{CN}2$, utilizando como conjunto de treinamento somente o subconjunto inicial de exemplos rotulados. Como conjunto de teste é utilizado o subconjunto, ou uma parte do subconjunto, restante dos exemplos. Assim, tanto nesse caso como no anterior, nenhum dos exemplos do conjunto de treinamento é utilizado no conjunto de teste.

Como citado anteriormente, para os conjuntos de dados *Breast–Cancer 2* e *Hepatitis*, nenhum exemplo foi rotulado pelo algoritmo utilizando como medida de similaridade a distância Chebychev. Isso deve-se ao fato desses dois conjuntos possuírem atributos discretos. No conjunto *Breast–Cancer 2*, por exemplo, dos seus 9 (nove) atributos, 5 (cinco) são discretos. Já no conjunto *Hepatitis*, dos seus 19 (dezenove) atributos, 13 (treze) possuem valores discretos.

Como descrito anteriormente, o algoritmo $k\text{-}means_{ki}$ foi concebido para tratar valores contínuos e discretos, bem como ausentes. Como o $k\text{-}means_{ki}$ trabalha com os dados normalizados, quando da comparação de dois atributos discretos utilizando a distância Chebychev (Equação 3.6 na página 34), é atribuído o valor 1 (um), a máxima distância possível, caso eles possuam valores diferentes. Isso faz com que o vetor v, responsável pela determinação do *threshold*, seja ocupado, quase que em todas as suas posições, por 1 (um). Por conseguinte, ao *threshold* $t^{'}$, na maioria dos casos, será também atribuído o valor 1 (um). Já que 1 (um) é a maior distância possível, e $t^{'} = 1$, obviamente todos os exemplos não rotulados, sem exceção, encontram-se a uma distância $\leq t^{'}$ de todos os clusters. Conseqüentemente, os exemplos não rotulados acabam pertencendo a clusters com classes diferentes e, então, não são rotulados pelo algoritmo.

A seguir são apresentadas as tabelas com os resultados obtidos com cada um dos conjuntos de dados utilizados.

6.3.1 *Conjunto* Breast–Cancer 2

Esse conjunto de dados consiste de 285 exemplos com 9 (nove) atributos, 5 (cinco) deles nominais. Nas tabelas 6.3, 6.5 e 6.7 são mostrados os exemplos rotulados pelo algoritmo utilizando um subconjunto inicial de exemplos rotulados contendo, respectivamente, 14 ($\sim 5\%$), 28 ($\sim 10\%$) e 56 ($\sim 20\%$) exemplos.

Como pode ser observado, a percentagem de exemplos rotulados pelo algoritmo (**% Rotulados**) é sempre maior quando a versão 2 é utilizada. Para entender esse fato, considere a Figura 6.3.

Nela, em (a) são ilustrados os exemplos inicialmente rotulados ($+$ e $*$), com seus respectivos clusters, e os não rotulados ($\bullet$). Em (b) é mostrado o resultado da primeira iteração do algoritmo $k\text{-}means_{ki}$. Como a versão 1 do algoritmo não atualiza os centróides quando da inclusão de exemplos nos clusters, em (b) é ilustrado o resultado da versão 1 do $k\text{-}means_{ki}$ aplicado sobre os dados em (a). Entretanto, a versão 2 possui a característica de atualizar os centróides sempre que há inclusão de novos exemplos em um cluster. Essa atualização é ilustrada em (c), (d) e (e), o que acarreta em um aumento do número de exemplos rotulados — de 11 (onze) para 20 (vinte). Conclui-se que a versão 2 tende a trazer o centro do cluster mais próximo do agrupamento, no caso dos exemplos inicialmente rotulados serem representativos.

Pode também ser observado que a percentagem de exemplos erroneamente rotulados pelo algoritmo (**% Errados**) é, na maioria dos casos, maior para a versão 2. Nesse caso, uma vez que um exemplo é rotulado erroneamente, ele tende a trazer o centro do cluster próximo de si, acarretando em novas rotulações erradas para aqueles exemplos

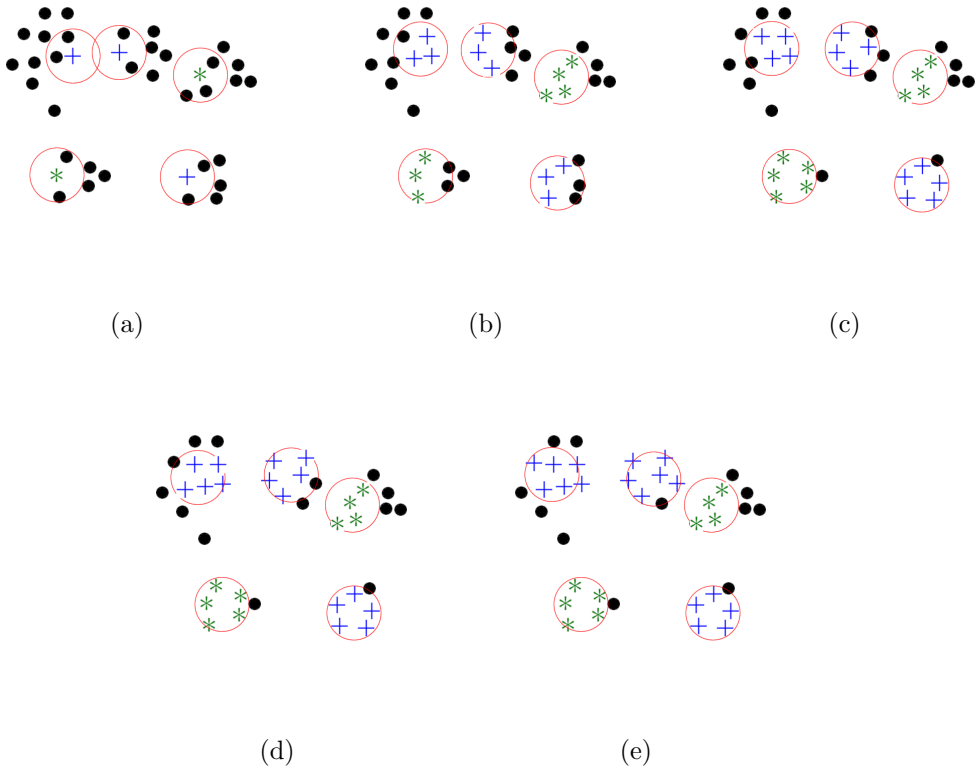


Figura 6.3: Versão 2 do algoritmo $k\text{-}means_{ki}$

não rotulados do seu agrupamento.

Um outro aspecto a ser observado é que, considerando os exemplos rotulados pelo algoritmo, $\frac{1}{3}$ (um terço) deles são rotulados erroneamente.

Já nas tabelas 6.4, 6.6 e 6.8 é mostrado o erro dos classificadores induzidos a partir do pequeno conjunto de exemplos inicialmente rotulados (**Pré-Rotulação**) e a partir desse mesmo conjunto adicionado dos exemplos rotulados pelo algoritmo $k\text{-}means_{ki}$ (**Pós-Rotulação**). É importante observar que tanto para esse conjunto de dados quanto para o conjunto de dados *Hepatitis* — Seção 6.3.2 na página 87 — o erro na coluna **Pré-Rotulação** foi calculado utilizando como conjunto de teste todos os exemplos que não pertencem ao conjunto de exemplos inicialmente rotulados. Ou seja, o conjunto de teste contém mais exemplos que o conjunto de teste utilizado para calcular o erro na coluna **Pós-Rotulação** dos classificadores induzidos utilizando o conjunto de exemplos inicialmente rotulados adicionado dos exemplos rotulados pelo $k\text{-}means_{ki}$.

Pode ser observado que na maioria dos casos os classificadores induzidos na fase pós-rotulação, utilizando como conjunto de teste todos os exemplos não rotulados pelo algoritmo, foram menos precisos que aqueles induzidos na fase pré-rotulação.

Como esperado, o erro desses classificadores é maior que o erro dos classificadores induzidos utilizando todos os exemplos — Tabela 6.2 na página 80.

Distância	*Threshold*	**Versão**	**Rotulados**			
			Total	**% Rotulados**	**Errados**	**% Errados**
Euclideana	5%	v1	104.8±31.8	38.7	31.4±5.9	30.0
		v2	144.8±41.7	53.4	44.6±9.0	30.8
	10%	v1	128.0±44.7	47.2	40.0±8.3	31.2
		v2	192.4±30.4	71.0	67.6±5.5	35.1
	20%	v1	107.0±36.2	39.5	35.2±6.6	32.9
		v2	167.2±35.4	61.7	58.4±11.3	34.9

Tabela 6.3: *Breast–Cancer 2* — Resultados com 14 ($\sim 5\%$) exemplos rotulados

Erro						
			Pós-Rotulação		**Pré-Rotulação**	
Distância	*Threshold*	**Versão**	$\mathcal{C}$4.5rules	$\mathcal{CN}2$	$\mathcal{C}$4.5rules	$\mathcal{CN}2$
Euclideana	5%	v1	36.9±5.3	34.7±5.6	37.7±9.2	33.6±4.1
		v2	39.7±5.3	38.3±4.7		
	10%	v1	40.9±4.8	42.0±4.9		
		v2	42.3±6.3	44.6±7.3		
	20%	v1	40.0±8.3	33.9±5.4		
		v2	41.4±4.9	35.4±7.6		

Tabela 6.4: *Breast–Cancer 2* — Performance dos classificadores antes (com 14 exemplos rotulados) e depois da rotulação

Distância	*Threshold*	**Versão**	**Rotulados**			
			Total	**% Rotulados**	**Errados**	**% Errados**
Euclideana	5%	v1	118.0±14.4	45.9	37.2±6.9	31.5
		v2	168.0±10.9	65.3	58.6±7.9	34.9
	10%	v1	107.6±21.2	41.9	32.8±6.4	30.5
		v2	172.4±33.0	67.1	54.0±7.9	31.3
	20%	v1	65.0±12.2	25.3	23.0±7.2	35.4
		v2	116.2±40.2	45.2	37.0±8.2	31.8

Tabela 6.5: *Breast–Cancer 2* — Resultados com 28 ($\sim 10\%$) exemplos rotulados

Erro						
Pós-Rotulação					**Pré-Rotulação**	
Distância	*Threshold*	**Versão**	$\mathcal{C}$4.5rules	$\mathcal{CN}2$	$\mathcal{C}$4.5rules	$\mathcal{CN}2$
Euclideana	5%	v1	38.7±5.0	34.7±6.3	33.8±6.4	31.8±2.1
		v2	45.0±2.0	39.8±4.0		
	10%	v1	39.3±6.8	32.5±3.4		
		v2	43.1±6.7	33.9±3.2		
	20%	v1	36.2±5.0	30.4±3.8		
		v2	33.6±8.6	31.1±7.2		

Tabela 6.6: *Breast–Cancer 2* — Performance dos classificadores antes (com 28 exemplos rotulados) e depois da rotulação

			Rotulados			
Distância	*Threshold*	**Versão**	**Total**	% **Rotulados**	**Errados**	% **Errados**
Euclideana	5%	v1	103.6±14.6	45.2	28.2±2.9	27.2
		v2	130.4±16.6	56.9	39.6±11.9	30.4
	10%	v1	76.4±22.1	33.4	21.2±8.6	27.7
		v2	93.8±21.8	41.0	36.2±6.4	38.6
	20%	v1	29.2±15.9	12.7	10.2±5.0	34.9
		v2	36.0±11.9	15.7	12.4±4.5	34.4

Tabela 6.7: *Breast–Cancer 2* — Resultados com 56 ($\sim$ 20%) exemplos rotulados

Erro						
Pós-Rotulação					**Pré-Rotulação**	
Distância	*Threshold*	**Versão**	$\mathcal{C}$4.5rules	$\mathcal{CN}2$	$\mathcal{C}$4.5rules	$\mathcal{CN}2$
Euclideana	5%	v1	35.1±5.8	30.7±6.0	30.4±2.2	29.6±2.6
		v2	37.6±5.1	31.0±5.1		
	10%	v1	30.6±0.6	29.1±2.0		
		v2	32.5±4.9	27.0±2.6		
	20%	v1	28.5±3.4	28.8±1.7		
		v2	30.8±1.0	27.1±1.8		

Tabela 6.8: *Breast–Cancer 2* — Performance dos classificadores antes (com 56 exemplos rotulados) e depois da rotulação

6.3.2 *Conjunto* Hepatitis

Esse conjunto de dados é composto de 155 exemplos com 19 (dezenove) atributos, sendo apenas 6 (seis) deles contínuos. Nas tabelas 6.9, 6.11 e 6.13 são mostrados os exemplos rotulados pelo algoritmo utilizando um subconjunto inicial de exemplos rotulados contendo, respectivamente, 7 ($\sim 5\%$), 15 ($\sim 10\%$) e 30 ($\sim 20\%$) exemplos.

Como pode ser observado, para esse conjunto de dados também a percentagem de exemplos rotulados pelo algoritmo k-$means_{ki}$ (**% Rotulados**) é sempre maior quando a versão 2 é utilizada. Nesse caso, a percentagem de exemplos erroneamente rotulados pelo algoritmo (**% Errados**) também é maior para a versão 2. Entretanto, essa percentagem é sempre menor que a obtida com o conjunto de dados *Breast–Cancer 2*, pois este conjunto de dados verifica melhor as premissas relacionadas ao espaço de exemplos nas quais sustenta-se o k-$means_{ki}$.

Um outro aspecto a ser observado é que em apenas um caso o aumento do *threshold* fez diminuir o número total de exemplos rotulados pelo algoritmo, quando comparado ao total de exemplos rotulados com um *threshold* menor.

Deve ser ressaltado que a taxa de erro do algoritmo k-$means_{ki}$ (**% Errados**) manteve praticamente a mesma variação.

Já nas tabelas 6.10, 6.12 e 6.14 é mostrado o erro dos classificadores induzidos a partir do pequeno conjunto de exemplos inicialmente rotulados (**Pré-Rotulação**) e a partir desse mesmo conjunto adicionado dos exemplos rotulados pelo algoritmo k-$means_{ki}$ (**Pós-Rotulação**). Ambos os erros foram calculados utilizando como conjunto de teste os conjuntos descritos anteriormente para o conjunto de dados *Breast–Cancer* — Seção 6.3.1 na página 83.

Pode ser observado que em todos os casos os classificadores induzidos na fase pós-rotulação, utilizando como conjunto de teste todos os exemplos não rotulados pelo algoritmo, foram menos precisos que aqueles induzidos na fase pré-rotulação.

Como esperado, o erro desses classificadores é maior que o erro dos classificadores induzidos utilizando todos os exemplos — Tabela 6.2 na página 80. Entretanto, pode ser observado que esse erro não é muito maior, como no caso do conjunto de dados *Breast–Cancer 2*. Como mencionado anteriormente, isso deve-se ao fato desse conjunto de exemplos verificar melhor as premissas nas quais sustenta-se o k-$means_{ki}$.

Distância	*Threshold*	**Versão**	**Rotulados**			
			Total	% **Rotulados**	**Errados**	% **Errados**
Euclideana	5%	v1	43.6±3.8	29.4	2.2±1.6	5.0
		v2	55.4±8.8	37.4	3.6±2.6	6.4
	10%	v1	69.0±6.0	46.6	6.2±2.3	9.0
		v2	86.0±11.1	58.1	11.0±4.2	12.8
	20%	v1	96.0±12.6	64.9	13.0±4.0	13.5
		v2	115.4±10.6	78.0	19.0±4.6	16.5

Tabela 6.9: *Hepatitis* — Resultados com 7 ($\sim 5\%$) exemplos rotulados

Erro						
Pós-Rotulação					**Pré-Rotulação**	
Distância	*Threshold*	**Versão**	$\mathcal{C}$4.5rules	$\mathcal{CN}2$	$\mathcal{C}$4.5rules	$\mathcal{CN}2$
Euclideana	5%	v1	24.7±3.2	28.0±7.6	20.9±0.0	22.8±5.9
		v2	25.8±2.9	28.0±7.3		
	10%	v1	31.6±10.2	27.3±5.6		
		v2	30.9±12.6	29.3±5.8		
	20%	v1	33.7±5.1	32.8±6.2		
		v2	32.1±9.0	35.7±9.0		

Tabela 6.10: *Hepatitis* — Performance dos classificadores antes (com 7 exemplos rotulados) e depois da rotulação

Distância	*Threshold*	**Versão**	**Rotulados**			
			Total	% **Rotulados**	**Errados**	% **Errados**
Euclideana	5%	v1	47.8±5.9	34.1	2.4±1.5	5.0
		v2	57.6±8.9	41.1	4.0±3.4	6.9
	10%	v1	74.0±12.0	52.8	8.6±2.9	11.6
		v2	96.4±12.2	68.8	16.0±4.7	16.6
	20%	v1	98.2±13.1	70.1	13.2±1.6	13.4
		v2	117.6±6.2	84.0	21.2±6.8	18.0

Tabela 6.11: *Hepatitis* — Resultados com 15 ($\sim 10\%$) exemplos rotulados

Erro						
Pós-Rotulação					**Pré-Rotulação**	
Distância	*Threshold*	**Versão**	$\mathcal{C}$4.5rules	$\mathcal{CN}2$	$\mathcal{C}$4.5rules	$\mathcal{CN}2$
Euclideana	5%	v1	28.3±4.3	25.6±3.3	20.6±5.9	19.6±3.9
		v2	29.1±4.4	28.7±3.8		
	10%	v1	36.0±5.4	29.8±3.6		
		v2	36.8±3.9	34.7±7.4		
	20%	v1	37.8±5.6	36.7±4.6		
		v2	37.1±20.2	32.7±11.1		

Tabela 6.12: *Hepatitis* — Performance dos classificadores antes (com 15 exemplos rotulados) e depois da rotulação

			Rotulados			
Distância	*Threshold*	**Versão**	**Total**	**% Rotulados**	**Errados**	**% Errados**
Euclideana	5%	v1	55.2±6.7	44.1	6.0±3.5	10.9
		v2	63.8±9.0	51.0	7.4±4.0	11.6
	10%	v1	75.2±12.3	60.1	11.0±3.0	14.6
		v2	87.4±13.3	69.9	15.8±1.6	18.1
	20%	v1	66.0±28.9	52.8	11.0±4.5	16.7
		v2	86.4±25.9	69.1	17.8±3.8	20.6

Tabela 6.13: *Hepatitis* — Resultados com 30 ($\sim$ 20%) exemplos rotulados

Erro						
Pós-Rotulação					**Pré-Rotulação**	
Distância	*Threshold*	**Versão**	$\mathcal{C}$4.5rules	$\mathcal{CN}2$	$\mathcal{C}$4.5rules	$\mathcal{CN}2$
Euclideana	5%	v1	29.3±5.9	25.3±6.2	19.8±2.7	21.9±4.6
		v2	30.5±4.5	27.9±7.6		
	10%	v1	29.9±11.6	24.8±9.0		
		v2	30.3±14.8	28.2±12.5		
	20%	v1	28.9±7.8	27.2±8.4		
		v2	26.8±12.9	25.3±12.0		

Tabela 6.14: *Hepatitis* — Performance dos classificadores antes (com 30 exemplos rotulados) e depois da rotulação

Ainda que os resultados obtidos com esse conjunto de exemplos, considerando o número de exemplos rotulados erroneamente pelo algoritmo, são superiores aos obtidos com o conjunto de exemplos *Breast–Cancer 2* — Seção 6.3.1 na página 83 — ,os

resultados deixam a desejar. Como explicado anteriormente, um dos motivos deve-se à função de distância utilizada para atributos discretos — Seção 5.2 na página 65 — a qual não manipula atributos discretos apropriadamente.

Uma forma de lidar com conjuntos de exemplos com atributos contínuos e discretos é utilizar uma função de distância heterogênea que utiliza funções de distância diferentes para diferentes tipos de atributos, as quais serão consideradas em trabalhos futuros.

O próximo conjunto de exemplos não apresenta esse problema já que possui somente atributos contínuos.

6.3.3 *Conjunto* Breast–Cancer

Esse conjunto de dados é composto de 699 exemplos com 9 (nove) atributos, sendo todos eles contínuos. Nas tabelas 6.15, 6.17 e 6.19 são mostrados os exemplos rotulados pelo algoritmo utilizando um subconjunto inicial de exemplos rotulados contendo, respectivamente, 34 ($\sim 5\%$), 69 ($\sim 10\%$) e 139 ($\sim 20\%$) exemplos.

Analogamente aos conjuntos já apresentados, para esse conjunto de dados a percentagem de exemplos rotulados pelo algoritmo (**% Rotulados**) é, na maioria dos casos, maior quando a versão 2 é utilizada. Pode também ser observado que a percentagem de exemplos erroneamente rotulados pelo algoritmo (**% Errados**) é, quando não nula, sempre maior para a versão 2. Também pode ser notado que conforme o *threshold* aumenta, a percentagem de exemplos rotulados pelo algoritmo também aumenta.

Pode ser observado que, contrariamente aos casos anteriores, o algoritmo consegue rotular novos exemplos utilizando a distância Chebychev. Entretanto, em todos os casos, a percentagem de exemplos rotulados erroneamente é sempre superior para a distância Chebychev. Isso mostra a influência da função de distância utilizada, como mencionado na Seção 3.3.1 na página 35.

Ainda assim, em ambos os casos, o número de exemplos rotulados erroneamente é muito baixo, mostrando a potencialidade do algoritmo proposto.

Já nas tabelas 6.16, 6.18 e 6.20 é mostrado o erro dos classificadores induzidos a partir do pequeno conjunto de exemplos inicialmente rotulados (**Pré-Rotulação**) e a partir desse mesmo conjunto adicionado dos exemplos rotulados pelo algoritmo $k\text{-}means_{ki}$ (**Pós-Rotulação**). Entretanto, para esse conjunto de dados que apresenta bons resultados, e com o objetivo de permitir uma comparação mais apropriada, foi utilizado para medir o erro **Pré-Rotulação** o mesmo conjunto de teste utilizado para medir o erro **Pós-Rotulação**.

Com o mesmo conjunto de teste, pode ser notado que, utilizando o conjunto de exemplos original adicionado do conjunto de exemplos rotulados pelo $k\text{-}means_{ki}$, na

maioria das vezes, praticamente manteve-se a taxa de erro dos classificadores induzidos, quando comparado à taxa de erro dos classificadores induzidos na fase **Pré-Rotulação**.

			Rotulados			
Distância	*Threshold*	**Versão**	**Total**	**% Rotulados**	**Errados**	**% Errados**
Euclideana	5%	v1	301.8±8.1	45.4	0	0
		v2	305.6±9.7	45.9	0	0
	10%	v1	352.2±19.0	53.0	0	0
		v2	358.4±20.6	53.9	0	0
	20%	v1	389.4±12.9	58.5	0.4±0.5	0.1
		v2	395.8±12.4	59.5	0.4±0.5	0.1
Chebychev	5%	v1	337.2±6.8	50.7	0	0
		v2	337.2±6.8	50.7	0	0
	10%	v1	395.0±36.8	59.4	3.2±2.2	0.8
		v2	403.0±40.0	60.6	4.2±3.0	1.0
	20%	v1	410.8±10.7	61.8	4.0±1.2	1.0
		v2	419.2±5.8	63.0	5.2±1.9	1.2

Tabela 6.15: *Breast–Cancer* — Resultados com 34 ($\sim 5\%$) exemplos rotulados

Erro						
Pós-Rotulação					**Pré-Rotulação**	
Distância	*Threshold*	**Versão**	$\mathcal{C}$4.5rules	$\mathcal{CN}2$	$\mathcal{C}$4.5rules	$\mathcal{CN}2$
Euclideana	5%	v1	13.1±0.7	13.5±2.1	17.5±1.0	17.6±2.4
		v2	13.3±0.8	13.6±2.2	17.6±1.1	17.8±2.4
	10%	v1	16.1±0.9	20.5±3.4	18.9±1.4	20.3±3.2
		v2	17.0±1.0	20.8±3.4	19.0±1.4	20.7±3.3
	20%	v1	20.5±2.2	20.2±2.1	18.4±2.5	22.0±3.8
		v2	23.2±2.9	24.3±4.3	18.3±2.7	22.3±3.9
Chebychev	5%	v1	17.6±1.1	14.6±2.5	18.3±1.4	19.4±2.5
		v2	17.6±1.1	14.6±2.5	18.3±1.4	19.4±2.5
	10%	v1	18.6±3.5	20.9±5.2	17.1±3.0	22.4±4.3
		v2	20.1±2.8	21.3±4.7	17.3±2.8	22.1±4.1
	20%	v1	17.4±3.6	21.9±5.0	17.8±3.2	22.9±4.2
		v2	18.8±3.0	22.3±4.5	17.9±2.9	22.6±3.9

Tabela 6.16: *Breast–Cancer* — Performance dos classificadores antes (com 34 exemplos rotulados) e depois da rotulação

			Rotulados			
Distância	*Threshold*	**Versão**	**Total**	**% Rotulados**	**Errados**	**% Errados**
Euclideana	5%	v1	313.6±6.1	49.8	0	0
		v2	317.6±5.5	50.4	0	0
	10%	v1	357.2±2.8	56.7	0	0
		v2	363.8±3.7	57.7	0	0
	20%	v1	383.6±2.7	60.9	0.4±0.5	0.1
		v2	387.2±2.2	61.5	0.8±0.8	0.2
Chebychev	5%	v1	347.2±3.8	59.4	0	0
		v2	347.2±3.8	59.4	0	0
	10%	v1	405.2±13.4	64.3	5.2±1.5	1.3
		v2	414.2±4.9	65.7	6.2±1.1	1.5
	20%	v1	405.2±13.4	64.3	5.2±1.5	1.3
		v2	414.2±4.9	65.7	6.2±1.1	1.5

Tabela 6.17: *Breast–Cancer* — Resultados com 69 ($\sim 10\%$) exemplos rotulados

Erro						
Pós-Rotulação					**Pré-Rotulação**	
Distância	*Threshold*	**Versão**	$\mathcal{C}$4.5rules	$\mathcal{CN}2$	$\mathcal{C}$4.5rules	$\mathcal{CN}2$
Euclideana	5%	v1	12.8±1.6	17.2±2.2	13.5±0.4	16.5±1.7
		v2	13.0±1.5	17.4±2.2	13.6±0.3	16.7±1.7
	10%	v1	17.6±3.0	18.0±2.5	15.0±0.6	19.2±1.9
		v2	18.8±3.3	18.7±2.5	15.2±0.7	19.6±2.0
	20%	v1	18.7±3.1	20.6±2.7	15.0±0.9	21.0±2.3
		v2	18.9±3.2	21.9±2.5	14.8±1.0	21.1±2.3
Chebychev	5%	v1	18.1±3.0	19.0±2.4	14.5±0.5	18.4±1.8
		v2	18.1±3.0	19.0±2.4	14.5±0.5	18.4±1.8
	10%	v1	14.0±0.9	21.9±1.4	13.9±0.8	19.7±2.6
		v2	14.1±0.8	20.1±2.7	13.8±0.7	19.7±2.7
	20%	v1	14.0±0.9	21.9±1.4	13.9±0.8	19.7±2.7
		v2	14.1±0.8	20.1±2.7	13.8±0.7	19.7±2.7

Tabela 6.18: *Breast–Cancer* — Performance dos classificadores antes (com 69 exemplos rotulados) e depois da rotulação

Distância	*Threshold*	**Versão**	**Rotulados**			
			Total	**% Rotulados**	**Errados**	**% Errados**
Euclideana	5%	v1	293.2±9.4	52.3	0	0
		v2	295.0±9.9	52.7	0	0
	10%	v1	325.6±4.8	58.1	0	0
		v2	329.4±5.0	58.8	0	0
	20%	v1	348.4±7.1	62.2	2.0±0.7	0.6
		v2	352.6±6.9	63.0	3.6±1.1	1.0
Chebychev	5%	v1	317.0±4.9	56.6	0.8±0.8	0.2
		v2	317.0±4.9	56.6	0.8±0.8	0.2
	10%	v1	387.2±7.3	69.1	7.4±1.5	1.9
		v2	392.8±6.4	70.1	8.0±1.7	2.0
	20%	v1	387.2±7.3	69.1	7.4±1.5	1.9
		v2	392.8±6.4	70.1	8.0±1.7	2.0

Tabela 6.19: *Breast–Cancer* — Resultados com 139 ($\sim 20\%$) exemplos rotulados

Distância	*Threshold*	**Versão**	**Erro**			
			Pós-Rotulação		**Pré-Rotulação**	
			$\mathcal{C}$4.5rules	$\mathcal{CN}2$	$\mathcal{C}$4.5rules	$\mathcal{CN}2$
Euclideana	5%	v1	11.0±0.9	13.9±1.5	11.2±1.1	14.7±1.2
		v2	11.4±1.2	14.5±1.2	11.2±1.1	14.7±1.2
	10%	v1	12.4±0.85	18.1±1.8	11.8±0.9	16.7±1.5
		v2	13.7±1.1	18.1±2.1	11.7±0.8	17.0±1.5
	20%	v1	15.1±1.2	19.3±2.2	11.5±0.9	17.7±1.7
		v2	12.3±1.3	20.7±2.0	11.1±0.9	17.4±1.8
Chebychev	5%	v1	11.1±0.4	15.4±1.4	11.4±0.9	15.8±1.3
		v2	11.1±0.4	15.4±1.4	11.4±0.9	15.8±1.3
	10%	v1	12.8±0.6	15.6±0.8	12.0±0.9	16.4±1.0
		v2	12.8±0.6	16.0±1.3	12.3±0.9	16.8±1.0
	20%	v1	12.8±0.6	15.6±0.8	12.0±0.9	16.4±1.0
		v2	12.8±0.6	16.0±1.3	12.3±0.9	16.8±1.0

Tabela 6.20: *Breast–Cancer* — Performance dos classificadores antes (com 139 exemplos rotulados) e depois da rotulação

Entretanto, os resultados apresentados nas tabelas que comparam a performance dos classificadores induzidos antes e depois da rotulação de novos exemplos pelo algoritmo proposto, referem-se às simulações nas quais têm-se absoluto controle sobre o conjunto de exemplos utilizado.

Vale lembrar que o erro dos classificadores foi medido repetindo o experimento 5

(cinco) vezes. No caso do erro na coluna **Pré-Rotulação** foi escolhida, em cada um dos 5 (cinco) experimentos, uma percentagem de exemplos rotulados. O classificador foi induzido utilizando esse conjunto como conjunto de treinamento e todos os outros exemplos como conjunto de teste. No caso do erro na coluna **Pós-Rotulação**, para cada uma das 5 (cinco) experiências realizadas na fase **Pré-Rotulação**, o conjunto de treinamento foi incrementado com os novos exemplos rotulados pelo algoritmo e todos os outros exemplos foram utilizados como conjunto de teste — Figura 6.2 na página 79. Como mencionado, somente para esse último conjunto de dados foi utilizado o mesmo conjunto de teste na fase de **Pré-Rotulação** e **Pós-Rotulação**.

Porém, em uma situação de uso real do algoritmo o procedimento a ser seguido é bem diferente. Nesse caso, inicialmente, o erro dos classificadores induzidos com os exemplos rotulados deve ser estimado, utilizando, por exemplo, *k–fold cross–validation* e, após a aplicação do algoritmo, os novos exemplos rotulados devem ser adicionados ao conjunto de treinamento. O erro dos classificadores induzidos com esse novo conjunto de treinamento deve ser estimado e, se for o caso, o processo deve ser repetido — Figura 5.6 na página 73.

Na Tabela 6.21 são mostrados os resultados obtidos realizando essa experiência real. Nessa experiência foi utilizado o conjunto *Breast–Cancer*, já que este apresentou os melhores resultados entre os conjuntos de exemplos utilizados.

Erro							
Parâmetros				**Pós-Rotulação**		**Pré-Rotulação**	
Distância	**Rotulados**	*Threshold*	**Versão**	$\mathcal{C}$4.5rules	$\mathcal{CN}2$	$\mathcal{C}$4.5rules	$\mathcal{CN}2$
Euclideana	69	20%	v2	1.5±0.5	2.4±0.7	7.1±3.2	8.6±3.8
Chebychev	69	20%	v2	1.0±0.5	1.8±0.6		
Euclideana	139	5%	v1	3.3±0.8	3.3±0.9	10.8±2.5	8.6±3.0
Chebychev	139	5%	v1	3.5±0.7	3.2±0.5		

Tabela 6.21: *Breast–Cancer* — Performance dos classificadores antes e depois da rotulação, utilizando 10–*fold cross–validation*

Nessa experiência foram escolhidos, para atuarem como conjunto de exemplos inicialmente rotulados, 69 ($\sim 10\%$) e 139 ($\sim 20\%$) exemplos do conjunto *Breast–Cancer*. Como citado anteriormente, em situações de uso real do algoritmo k-$means_{ki}$, inicialmente deve ser induzido um classificador com o conjunto de exemplos inicialmente rotulados, e estimar sua precisão com, por exemplo, *k–fold cross–validation*. Nessa experiência foi utilizado 10–*fold cross-validation*, e os resultados obtidos são ilustrados na coluna **Pré-Rotulação** da Tabela 6.21.

O próximo passo da experiência consistiu na execução do k-$means_{ki}$ utilizando,

como conjunto de exemplos inicialmente rotulados os 69, e 139 exemplos utilizados na indução do classificador. Como conjunto de exemplos não rotulados foi utilizado o resto dos exemplos, *i.e.*, 630 e 560 exemplos respectivamente, visto que o conjunto *Breast–Cancer* possui 699 exemplos.

Para o primeiro caso — 69 exemplos no conjunto de exemplos inicialmente rotulados — , o k-$means_{ki}$ foi executado utilizando um *threshold* de 20% com a versão 2 do algoritmo. Já no segundo caso — 139 exemplos no conjunto de exemplos inicialmente rotulados — , foi utilizado um *threshold* de 5% com a versão 1 do algoritmo[3]. Para ambos os casos foram utilizadas as medidas Chebychev e Euclideana.

Após a execução do k-$means_{ki}$, para analisar os rótulos encontrados, foram induzidos classificadores utilizando o conjunto de exemplos inicialmente rotulados adicionado do conjunto de exemplos rotulados pelo algoritmo. Esses classificadores também tiveram sua precisão estimada utilizando 10–*fold cross–validation*, a qual é mostrada na coluna **Pós-Rotulação** da Tabela 6.21.

Na Tabela 6.22 é mostrado o número de exemplos rotulados pelo algoritmo em cada um dos casos, bem como a percentagem de exemplos rotulados considerando, para cada caso, o total de exemplos não rotulados.

Erro						
Parâmetros				**Total Rotulados**	**% Total**	**Errados**
Distância	**Rotulados**	*Threshold*	**Versão**			
Euclideana	69	20%	v2	389	61.7	0
Chebychev	69	20%	v2	419	66.5	8
Euclideana	139	5%	v1	309	55.2	0
Chebychev	139	5%	v1	325	58.0	0

Tabela 6.22: *Breast–Cancer* — Número de exemplos rotulados pelo k-$means_{ki}$

Por exemplo, utilizando a distância Euclideana com 69 exemplos inicialmente rotulados, o algoritmo rotulou 389 exemplos, os quais representam 61.7% dos 630 exemplos (*i.e.*, $699 - 69$) não rotulados.

Como pode ser observado, em três dos quatro casos nenhum exemplo foi erroneamente rotulado pelo k-$means_{ki}$.

Na Tabela 6.21 pode-se verificar que o erro dos classificadores induzidos na fase **Pós-Rotulação** é bem menor que os daqueles induzidos na fase **Pré-Rotulação**, para ambos os indutores utilizados no experimento. Assim, o k-$means_{ki}$ estaria apto a ser executado novamente, uma vez que os exemplos por ele rotulados foram capazes de

[3]Os valores de exemplos inicialmente rotulados, *threshold* e versão do algoritmo foram escolhidos, para essa experiência, de forma aleatória.

aumentar a precisão do classificador. Nesse caso, o conjunto de exemplos rotulados pelo $k\text{-}means_{ki}$ adicionado dos exemplos inicialmente rotulados seria, nessa nova iteração do algoritmo, fornecido ao $k\text{-}means_{ki}$ como sendo o conjunto inicial de exemplos rotulados, e apenas os exemplos não rotulados pelo algoritmo na iteração anterior pertenceriam ao conjunto dos exemplos não rotulados.

Como mencionado na Seção 6.2 na página 76, nas simulações realizadas, a escolha inicial dos exemplos rotulados foi feita de duas formas diferentes,

1. aleatoriamente e
2. utilizando o algoritmo de *clustering* probabilístico AUTOCLASS.

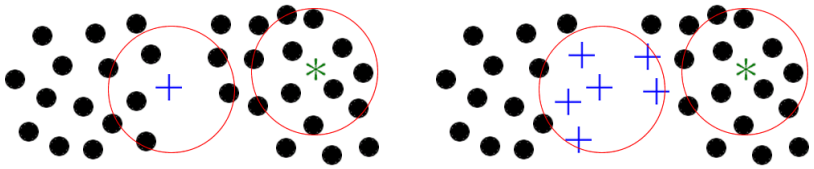
Os resultados obtidos utilizando AUTOCLASS, descritos em Sanches & Monard (2003), foram todos piores dos obtidos utilizando seleção aleatória de exemplos. Deve ser lembrado que AUTOCLASS é um algoritmo probabilístico. Assim, exemplos são agrupados em clusters considerando a probabilidade de um exemplo pertencer a um determinado cluster. Dessa forma, exemplos que pertencem a um mesmo cluster segundo AUTOCLASS, não se encontram, necessariamente, próximos segundo as medidas de distância utilizadas pelo $k\text{-}means_{ki}$, o que mostra a importância do conjunto inicial de exemplos rotulados para um bom desempenho do algoritmo proposto.

6.4 Importância do Conjunto de Exemplos Inicialmente Rotulados

Como citado anteriormente, a técnica utilizada pelo algoritmo proposto está baseada em duas premissas. Uma das premissas diz que a maioria dos exemplos nos clusters pertencem a uma classe específica. A outra premissa é que os exemplos tendem a se agrupar naturalmente em clusters, ao invés de se distribuirem uniformemente no espaço de descrição dos exemplos. Além disso, cada exemplo do conjunto inicial de exemplos rotulados deve estar localizado perto do centro de um dos clusters existentes no espaço de descrição de exemplos.

Deve ser observado que se essas premissas não são satisfeitas, os resultados do algoritmo estarão comprometidos. Por exemplo, um exemplo rotulado localizado distante do centro do agrupamento ao qual ele pertence pode acarretar, durante o processo de rotulação, em rótulos errados, uma vez que facilmente ele poderá rotular exemplos de outros agrupamentos próximos ao dele. Para ilustrar esse fato considere a Figura 6.4 na próxima página. Em (a) são ilustrados os exemplos inicialmente rotulados ($+$ e $*$) e os não rotulados ($\bullet$), e dois agrupamentos naturais podem ser notados, um à esquerda,

e outro à direita. Em (b) é mostrado o resultado da primeira iteração do algoritmo $k\text{-}means_{ki}$. É possível notar a rotulação errônea de dois exemplos do agrupamento da direita, por um exemplo pertencente ao agrupamento da esquerda.



(a) (b)

Figura 6.4: Conseqüência da utilização de exemplos pouco representativos

Ao se escolher aleatoriamente os exemplos a constituirem o subconjunto de exemplos inicialmente rotulados — estratégia adotada nos experimentos descritos anteriormente — não é garantido que estes estejam perto do centro do agrupamento do qual eles fazem parte. Ainda assim, os resultados obtidos com o conjunto de dados *Breast–Cancer* foram bons, o que aparentemente mostra que alguns exemplos não muito perto dos centros dos clusters são bem tolerados pelo algoritmo.

Com o objetivo de ilustrar a importância dos exemplos inicialmente rotulados para um bom desempenho do algoritmo, foi realizado um experimento com o conhecido conjunto de exemplos *Iris*. Nesse caso, primeiramente, os exemplos inicialmente rotulados foram escolhidos de forma a estarem localizados perto do centro de um agrupamento. Para se atingir esse objetivo, foi calculado o ponto médio das três classes encontradas no conjunto *Iris*, e escolheu-se, de cada classe, os dois pontos mais próximos do respectivo ponto médio. Os resultados obtidos, bem como uma breve descrição do conjunto *Iris* são apresentados a seguir.

6.4.1 *Conjunto* Iris

Esse conjunto de dados, que pode ser encontrado no repositório da UCI (Merz & Murphy, 1998), é composto por medidas (comprimento e largura) de sépalas e pétalas de 3 (três) diferentes tipos da planta *Iris*. É uma das bases de dados mais conhecidas e utilizadas na área de reconhecimento de padrões.

Esse conjunto contém 3 (três) classes (setosa, versicolor e virginica), com 50 exemplos de cada classe, no qual cada classe referencia um tipo de planta *Iris*. A primeira classe é linearmente separável das outras duas, enquanto as duas últimas não são linearmente separáveis uma da outra. Cada exemplo é descrito por 4 (quatro) atributos contínuos mais a classe correspondente.

Na Figura 6.5 é ilustrado esse conjunto de exemplos. Nela, os exemplos pertencentes à classe virginica são representados por ○, os exemplos pertencentes à classe versicolor por + e os exemplos pertencentes à classe setosa são representados por △. Já os exemplos rotulados escolhidos de cada classe são ilustrados, em vermelho, na Figura 6.6.

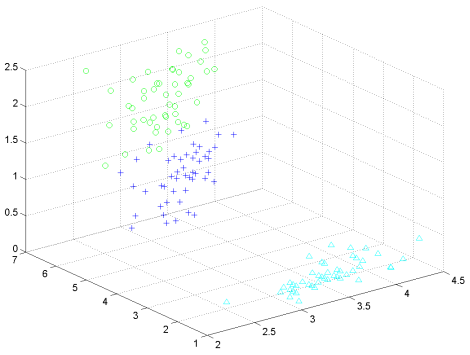


Figura 6.5: Conjunto *Iris*

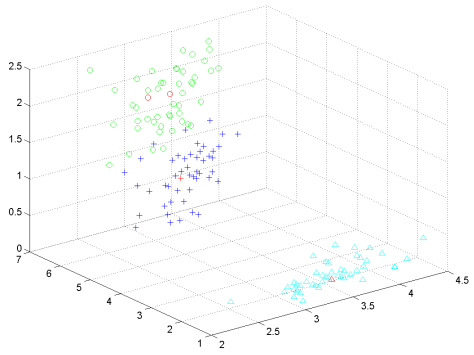


Figura 6.6: Conjunto *Iris* — Exemplos rotulados escolhidos

6.4.2 Resultados Obtidos

Na Tabela 6.23 são mostrados os resultados obtidos utilizando 6 (seis) exemplos rotulados, 2 (dois) de cada classe, que se encontram perto do centro dos clusters correspondentes a cada uma das 3 (três) classes do conjunto de dados *Iris*.

O experimento foi realizado somente uma vez já que se trata de um exemplo ilustrativo, e os resultados obtidos são ilustrados na Tabela 6.23 na página oposta. Como pode ser observado, os resultados obtidos são muito bons.

Na Tabela 6.24 é mostrado o erro dos classificadores induzidos nas simulações realizadas, antes e após a rotulação. Como pode ser observado, apesar de utilizar somente 6 (seis) exemplos iniciais rotulados, por serem eles exemplos muito especiais que se encontram perto do centro dos clusters, os classificadores induzidos conseguem, exceto em um caso, classificar razoavelmente bem os exemplos do conjunto de teste.

Nas figuras a seguir é ilustrado graficamente o processo de rotulação ao qual se refere a Tabela 6.23. Deve ser ressaltado que os gráficos levam em consideração apenas a distância Euclideana.

Na Figura 6.7(a) são mostrados, em vermelho, os exemplos rotulados pela versão 1 do $k\text{-}means_{ki}$, utilizando um *threshold* de 5%. Como descrito anteriormente, a versão 1 consiste de uma única iteração. Já na Figura 6.7(b) pode ser visto, também em vermelho, o resultado do processo de rotulação da versão 2 do $k\text{-}means_{ki}$, com o mesmo

Distância	*Threshold*	**Versão**	**Rotulados**		
			Total	**% Rotulados**	**Errados**
Euclideana	5%	v1	31	21.5	0
		v2	31	21.5	0
	10%	v1	54	37.5	0
		v2	59	41.0	0
	20%	v1	99	68.7	1
		v2	114	79.2	4
Chebychev	5%	v1	39	27.1	0
		v2	41	28.5	0
	10%	v1	65	45.1	0
		v2	67	46.5	0
	20%	v1	99	68.7	1
		v2	104	72.2	1

Tabela 6.23: *Iris* — Resultados com 6 (4%) exemplos rotulados apropriados para o *bias* do k-$means_{ki}$

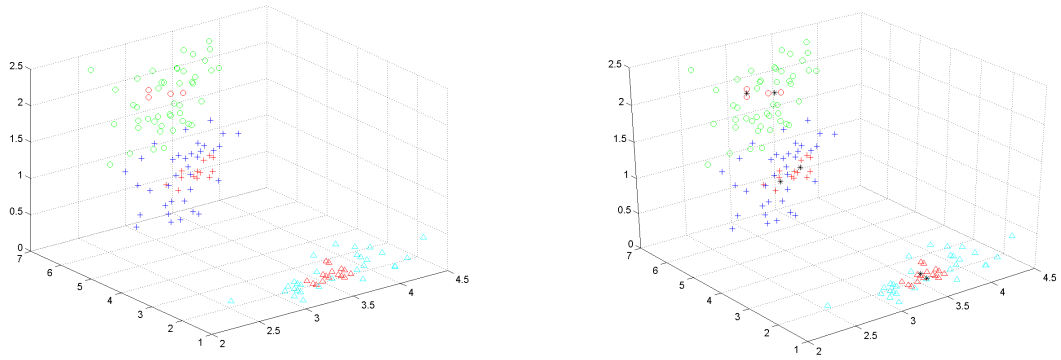
Erro no 5-cv						
Pós-Rotulação					**Pré-Rotulação**	
Distância	*Threshold*	**Versão**	$\mathcal{C}$4.5rules	$\mathcal{CN}2$	$\mathcal{C}$4.5rules	$\mathcal{CN}2$
Euclideana	5%	v1	8.85	7.96	14.58	9.03
		v2	8.85	7.96		
	10%	v1	11.11	5.56		
		v2	11.76	2.35		
	20%	v1	11.11	15.56		
		v2	20.00	20.00		
Chebychev	5%	v1	9.52	4.76		
		v2	9.71	4.85		
	10%	v1	11.39	2.53		
		v2	11.69	1.3		
	20%	v1	11.11	13.33		
		v2	12.50	14.50		

Tabela 6.24: *Iris* — Performance dos classificadores antes (com 6 exemplos rotulados apropriados para o *bias* do k-$means_{ki}$) e depois da rotulação

threshold de 5%. Os pontos representados por $*$ ilustram a posição do centróide na versão 2. Pode-se notar, nesse caso, tanto pelos gráficos quanto pelos resultados na Tabela 6.23, que a versão 2 do algoritmo comportou-se da mesma maneira que a versão 1, ambas rotulando 31 exemplos[4] corretamente.

Na Figura 6.8(a) são mostrados, em vermelho, os exemplos rotulados pela versão 1 do k-$means_{ki}$, utilizando um *threshold* de 10%. O resultado da versão 2 do algoritmo, utilizando o mesmo *threshold* de 10%, é ilustrado na Figura 6.8(b), também em vermelho. Os pontos representados por $*$ ilustram a posição do centróide na versão 2.

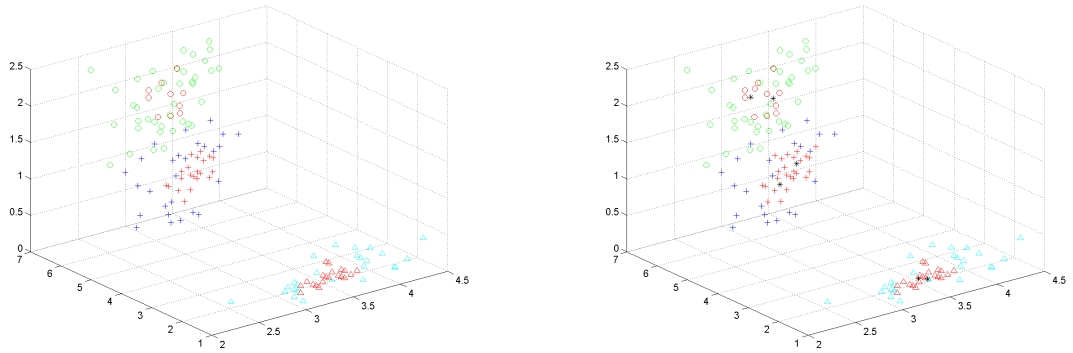
[4]Obviamente, os exemplos rotulados pela versão 1 são os mesmos que aqueles rotulados pela versão 2.



(a) (b)

Figura 6.7: Conjunto *Iris* aplicado ao *k-means*$_{ki}$ — *Threshold* 5%

Nesse caso, a versão 1 rotulou 54 exemplos e a versão 2 rotulou 59 exemplos. Em ambos os casos todos os exemplos foram rotulados corretamente.

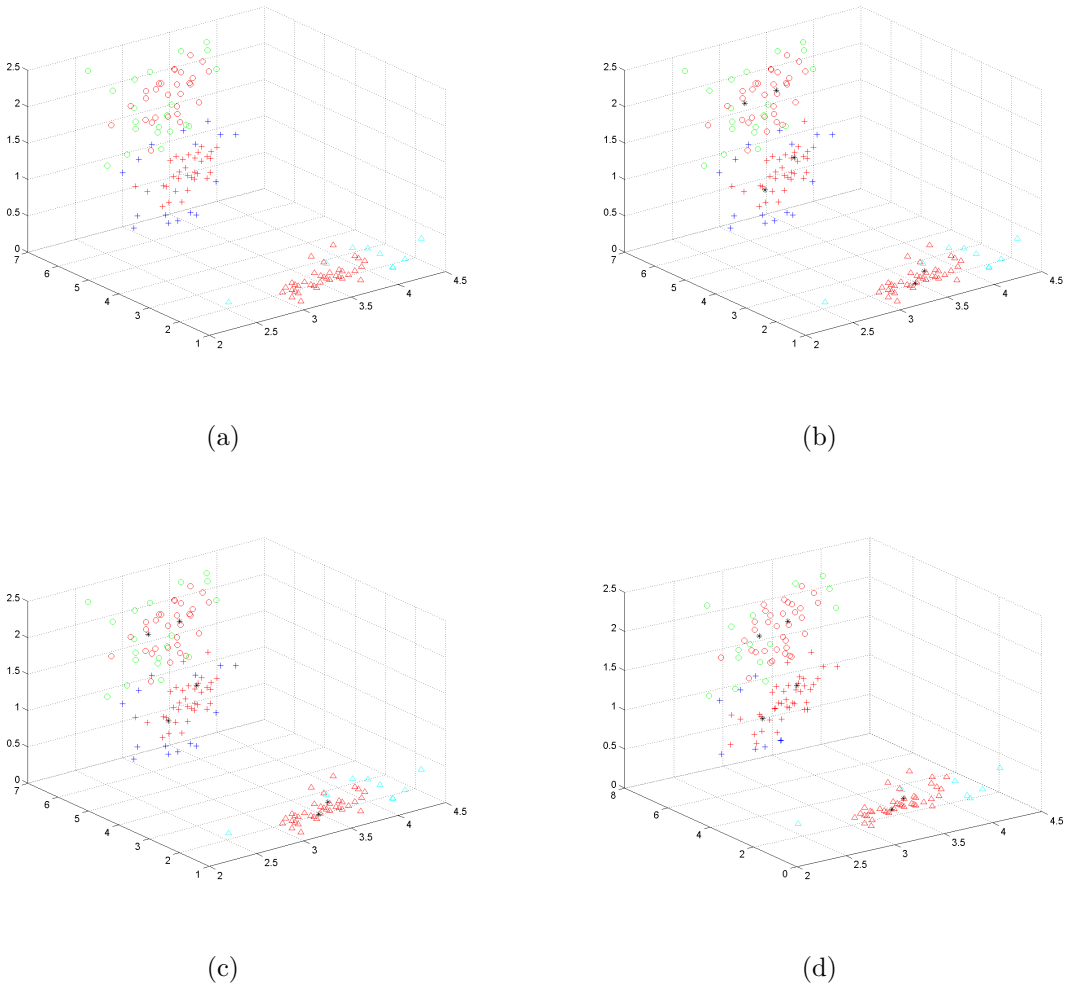


(a) (b)

Figura 6.8: Conjunto *Iris* aplicado ao *k-means*$_{ki}$ — *Threshold* 10%

Na Figura 6.9 é mostrado o resultado do *k-means*$_{ki}$ quando aplicado um *threshold* de 20%. Em (a) são mostrados, em vermelho, os exemplos rotulados pela versão 1 do *k-means*$_{ki}$. Em (b), (c) e (d) são ilustradas, respectivamente, a segunda, terceira e quarta iterações da versão 2 do algoritmo (vale lembrar que a primeira iteração da versão 2 corresponde à versão 1), estando os pontos em vermelho representando os exemplos rotulados pelo algoritmo. Os pontos representados por $*$ ilustram a posição do centróide em cada iteração da versão 2. Nessa experiência, a versão 1 rotulou 99 exemplos, sendo apenas 1 (um) exemplo rotulado erroneamente, e a versão 2 rotulou

114 exemplos, dos quais 4 (quatro) deles erroneamente.



(a) (b)

(c) (d)

Figura 6.9: Conjunto *Iris* aplicado ao k-$means_{ki}$ — *Threshold* 20%

6.5 Considerações Finais

Neste capítulo foram mostrados os resultados experimentais obtidos com o algoritmo k-$means_{ki}$ utilizando os conjuntos de dados *Breast–Cancer 2*, *Hepatitis* e *Breast–Cancer*. Esses conjuntos foram particionados em 2 (dois) subconjuntos, um pequeno, com exemplos rotulados e um grande, com não rotulados. Dessa forma, é simulada a situação para a qual o algoritmo k-$means_{ki}$ foi concebido, aquela em que está disponível um pequeno conjunto de exemplos rotulados e uma grande quantidade de exemplos não rotulados.

Os resultados mostram que o comportamento do algoritmo $k\text{-}means_{ki}$ é bom, desde que se verifiquem as premissas nas quais ele se apóia.

Para ilustrar a importância dos exemplos inicialmente rotulados para um bom desempenho do $k\text{-}means_{ki}$, foi utilizado o conhecido conjunto *Iris*, no qual os exemplos inicialmente rotulados foram escolhidos de forma a estarem localizados perto do centro dos respectivos agrupamentos.

Capítulo

7

Conclusões

"Diga a verdade, e saia correndo."

Anônimo

Neste trabalho é proposto um algoritmo, denominado $k\text{-}means_{ki}$, a ser utilizado no processo de rotulação de exemplos a partir de poucos exemplos rotulados. Baseado no conhecido algoritmo *k-means*, o algoritmo aqui proposto utiliza técnicas de clusterização para melhorar o processo de classificação, e está baseado em duas premissas. A primeira delas é que os exemplos tendem a se agrupar naturalmente em clusters, ao invés de se distribuirem uniformemente no espaço de descrição dos exemplos. Além disso, cada exemplo do conjunto inicial de exemplos rotulados deve estar localizado perto do centro de um dos clusters existentes no espaço de descrição de exemplos. A segunda premissa diz que a maioria dos exemplos nos clusters pertencem a uma classe específica.

Por trabalhar simultaneamente com exemplos rotulados e não rotulados, o $k\text{-}means_{ki}$ enquadra-se na categoria dos algoritmos de aprendizado semi-supervisionado.

A motivação para o seu uso deve-se ao fato de, na grande maioria das vezes, estar

disponível um pequeno conjunto de exemplos rotulados e uma grande quantidade de exemplos não rotulados. Essa situação, geralmente, inviabiliza o uso de algoritmos de aprendizado supervisionado. A idéia então consiste em incrementar, a partir da rotulação de novos exemplos, o conjunto de exemplos rotulados, procurando viabilizar a utilização de algoritmos de aprendizado supervisionado.

7.1 Resultados Iniciais

A maioria das conclusões obtidas com este trabalho podem ser feitas a partir dos resultados experimentais obtidos com o algoritmo k-$means_{ki}$. Essas conclusões são listadas a seguir.

- A qualidade dos exemplos inicialmente rotulados é de suma importância para o bom funcionamento do algoritmo proposto. O algoritmo k-$means_{ki}$, como citado anteriormente, apóia-se em duas premissas, ambas relacionadas com o conjunto de dados utilizado. Uma delas diz que os exemplos rotulados devem estar localizados perto do centro do agrupamento ao qual eles pertencem. Como foi mostrado — Seção 6.4 na página 96 — , exemplos pertencentes ao conjunto inicial de exemplos rotulados localizados longe do centro do agrupamento ao qual eles pertencem, podem, com grande probabilidade, acarretar na rotulação errônea de exemplos, o que não é desejável.

- A medida de distância utilizada, principalmente quando da utilização de conjuntos de dados heterogêneos (com atributos discretos e contínuos), influi, drasticamente no processo de rotulação. Como mostrado nos experimentos, utilizando os conjuntos de dados *Breast–Cancer 2* e *Hepatitis* — Seção 6.3.1 na página 83 e Seção 6.3.2 na página 87, respectivamente — com a distância Chebychev, o algoritmo k-$means_{ki}$ não rotulou nenhum exemplo, visto que essa distância não se mostrou apropriada para conjuntos de dados heterogêneos. Assim, ao se escolher a medida de similaridade a ser utilizada no processo de rotulação deve-se levar em conta os dados utilizados.

- O valor do *threshold* deve ser utilizado com cautela. A utilização de um alto *threshold* acarretará na formação de grandes clusters. Por conseguinte, aumenta-se a possibilidade de rotular um exemplo erroneamente, visto que quanto maior o *threshold*, mais longe o exemplo não rotulado pode estar do centro do cluster para ser rotulado. Como mostrado anteriormente — Seção 5.2 na página 65 — , o k-$means_{ki}$ utiliza restrições para rotular um exemplo, o que não necessariamente

diminuirá a percentagem de exemplos rotulados erroneamente. Além disso, tais restrições somente são válidas quando há uma intersecção de clusters originados por exemplos de classes diferentes. Caso contrário, o aumento do *threshold*, certamente acarretará no aumento do número de exemplos rotulados erroneamente.

- As duas versões propostas do algoritmo k-$means_{ki}$ comportam-se de forma semelhante. Pode ser comprovado pelas experiências realizadas que, na grande maioria dos casos, a versão 2 do k-$means_{ki}$ rotulou mais exemplos que a versão 1. Pode ser visto também que o erro no processo de rotulação foi maior para a versão 2. Conclui-se então que ao se escolher a versão a ser utilizada deve-se ponderar entre um número menor, mas com maior precisão, de exemplos rotulados — versão 1 — ou mais exemplos rotulados, com a possibilidade de aumentar o erro — versão 2.

Os resultados obtidos, utilizando o conjunto de dados *Breast–Cancer*, mostram que a idéia utilizada pelo algoritmo é promissora, provendo uma gama de trabalhos futuros, os quais são descritos a seguir.

7.2 Trabalhos Futuros

Implementação de medidas de similaridade heterogênas. O k-$means_{ki}$ possui dois módulos de medidas de similaridade implementados — distância Euclideana e distância Chebychev. A distância Euclideana é amplamente utilizada e bastante apropriada para atributos contínuos mas, freqüentemente, essa função de distância não manipula atributos discretos apropriadamente. Já a distância Chebychev é apropriada para os casos que exemplos são considerados diferentes pelo simples fato de possuirem um atributo diferente.

O algoritmo k-$means_{ki}$, nos moldes como está implementado hoje, utiliza uma abordagem simplista no tratamento de atributos discretos, atribuindo 1 (um), *i.e.*, a máxima distância entre dois atributos, quando estes são discretos e diferentes. Essa abordagem falha por não utilizar informações adicionais providas pelos valores de um atributo qualitativo, informações essas que poderiam ajudar no processo de aprendizado.

Uma métrica que tenta fazer uso de tais informações é a VDM (*Value Difference Metric*) (Stanfill & Waltz, 1986). A métrica VDM considera a similaridade de classificação entre cada possível valor de um atributo para calcular a distância entre esses valores. Ela considera dois valores similares se eles possuem classificações similares, isto é, se eles possuem correlações similares com a classe. Sendo assim,

torna-se interessante implementar a métrica VDM, bem como outras medidas de similaridade para serem utilizadas no k-$means_{ki}$ e avaliar o seu comportamento diante de atributos discretos.

Utilização da teoria de fractais. Há estudos no intuito de utilizar a dimensão fractal de Hausdorf D_0 para determinar a similaridade entre os exemplos (de Sousa, 2003). A idéia principal da técnica, proposta por Barbará & Chen (2000), é adicionar pontos do conjunto de dados aos agrupamentos já existentes tomando como base a alteração que esses pontos causam na dimensão fractal dos agrupamentos. Um determinado ponto é inserido no cluster em que a inserção causar a menor alteração na dimensão fractal.

Realização de experimentos com o auxílio de um especialista. Uma outra sugestão para trabalhos futuros é a realização de experimentos utilizando conjuntos de dados com nenhum exemplo rotulado. Assim, a idéia consiste na utilização de um especialista do domínio, responsável por rotular alguns exemplos muito representativos. Ao se utilizar, na fase Pós-Rotulação, classificadores simbólicos, espera-se que o especialista seja capaz de analisar os classificadores induzidos, buscando validar o processo de rotulação.

O uso do especialista é especialmente apropriado para quando do uso do k-$means_{ki}$ em aplicações médicas, e quaisquer outras aplicações nas quais um rótulo errado poderia trazer conseqüências desastrosas.

Planejamento de novas formas de união dos centróides iniciais. Como citado na Seção 5.2.3 na página 68, o algoritmo k-$means_{ki}$, nas suas versões 3 e 4 une os centróides iniciais que estão muito próximos, resultando em um único cluster. Entretanto, essa união produziu clusters muito grandes, e o processo de rotulação não foi bom. O processo de união proposto nas versões 3 e 4 consiste basicamente de um cluster resultante que englobe os dois clusters a serem unidos. Essa abordagem, um tanto quanto simples, não produziu resultados satisfatórios. Assim, torna-se interessante o estudo, e implementação, de novas formas de união de clusters junto ao k-$means_{ki}$, para analisar o seu comportamento.

Comparação do k-$means_{ki}$ com outros métodos de rotulação. Está em desenvolvimento no LABIC um projeto de mestrado (Matsubara, 2003), cujo objetivo consiste da pesquisa, projeto e implementação de algoritmos de aprendizado semi-supervisionado utilizando *Co-Training* com uma e duas visões — Seção 4.5.4 na página 56 e Seção 4.5.1 na página 53, respectivamente.

Assim, pretende-se comparar os resultados obtidos utilizando *Co-Training* com uma visão, a ser implementado nesse projeto, com os resultados obtidos pelo algoritmo k-$means_{ki}$, efetuando assim um estudo comparativo desses dois métodos de rotulação.

Pré-processamento nos dados. Como mencionado anteriormente, nos experimentos realizados neste trabalho não foram utilizadas técnicas de pré-processamento de dados para diminuir a dimensionalidade. Assim, uma possível sugestão de trabalho futuro seria a utilização do k-$means_{ki}$ após um pré-processamento dos dados, e comparar os resultados com aqueles obtidos com os dados originais, *i.e.*, utilizando todos os atributos.

Uma das técnicas utilizadas em pré-processamento de dados para *clustering* é o PCA — *Principal Component Analysis* — , que busca representar o conjunto de dados original por um número reduzido de atributos que retenham muitas das informações intrínsecas contidas nos dados. Em outras palavras, o conjunto de dados passa por uma redução de dimensionalidade.

Utilização de outros indutores. Neste trabalho foram utilizados os indutores $\mathcal{C}4.5$rules e $\mathcal{CN}2$ na fase Pós-Rotulação. Um interessante trabalho futuro seria a realização de experimentos com diferentes indutores e a posterior análise dos resultados obtidos.

Lista de Siglas

AM	Aprendizado de Máquina
DLE	Discover Learning Environment
DOL	Discover Object Library
DSX	Discover Dataset Sintax
EM	Expectation-Maximization
IA	Inteligência Artificial
IC	Indução Construtiva
KDD	Knowledge Discovery in Databases
LABIC	Laboratório de Inteligência Computacional
MD	Mineração de Dados
MT	Mineração de Textos
NIPS	Neural Information Processing Systems
PCA	Principal Component Analysis
SSA	Seleção de um Subconjunto de Atributos
SVM	Support Vector Machines
UCI	University of California, Irvine

Referências Bibliográficas

Aha, D. W., D. Kibler, & M. Albert (1991). Instance-based learning algorithms. *Machine Learning 6*, 37–66. 14

Baranauskas, J. A. (2001). Extração automática de conhecimento por múltiplos indutores. Tese de Doutorado, ICMC–USP. http://www.teses.usp.br/teses/disponiveis/55/55134/tde-08102001-112806/restrito/tese.pdf. 6

Baranauskas, J. A. & G. E. A. P. A. Batista (2000). O Projeto DISCOVER: Idéias iniciais. Comunicação pessoal. 5

Baranauskas, J. A. & M. C. Monard (1998). Experimental feature selection using the wrapper approach. In *Proceedings of the International Conference on Data Mining — ICDM'98*, Rio de Janeiro, RJ, pp. 161–170. http://www.fmrp.usp.br/~augusto/. 29

Baranauskas, J. A. & M. C. Monard (1999). The $\mathcal{MLC}$++ wrapper for feature subset selection using decision tree, production rule, instance based and statistical inducers: Some experimental results. Technical Report 87, ICMC–USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_87.ps.zip. 29

Baranauskas, J. A. & M. C. Monard (2000). An unified overview of six supervised symbolic machine learning inducers. Technical Report 103, ICMC–USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_103.ps.zip. 80

Barbará, D. & P. Chen (2000). Using the fractal dimension to cluster datasets. In R. Ramakrishnan, S. Stolfo, R. Bayardo, & I. Parsa (Eds.), *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining — KDD-00*, New York, USA, pp. 260–264. ACM Press. 106

Basu, S., A. Banerjee, & R. Mooney (2002). Semi-supervised clustering by seeding. In *Proceedings of the Nineteenth International Conference on Machine Learning — ICML-2002*, Sidney, Australia, pp. 19–26. 50, 51, 52

Batista, G. E. A. P. A. (1997). Um ambiente de avaliação de algoritmos de aprendizado de máquina utilizando exemplos. Dissertação de Mestrado, ICMC–USP. 6

Batista, G. E. A. P. A. (2002). Definição da Sintaxe DSX. http://www.icmc.usp.br/~gbatista/Discover/SintaxePadraoFinal.html. 71

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-19082002-234842. 6

Batista, G. E. A. P. A. & M. C. Monard (2003a). 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. 6, 72

Batista, G. E. A. P. A. & M. C. Monard (2003b). The Discover Object Library (DOL) User's Manual. Technical report, ICMC–USP. (in press). 6, 71

Bennett, K. & A. Demiriz (1999). Semi-supervised support vector machines. In S. A. S. M. S. Kearns & D. A. Cohn (Eds.), *Advances in Neural Information Processing Systems — NIPS*, Volume 11, Cambridge, MA, USA, pp. 368–374. MIT Press. 60

Bennett, K., A. Demiriz, & R. Maclin (2002). Exploiting unlabeled data in ensemble methods. In D. K. David Hand & R. Ng (Eds.), *Proceedings of the Eighth ACM SIGKDD Internation Conference on Knowledge Discovery and Data Mining*, Edmonton, Canadá, pp. 289–296. ACM Press. 59

Bernardini, F. C. (2002). Combinação de classificadores para melhorar o poder preditivo e descritivo de *ensembles*. Dissertação de Mestrado, ICMC–USP. 6

Blum, A. & S. Chawla (2001). Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of Eighteenth International Conference on Machine Learning — ICML-2001*, Williamstown, MA, USA, pp. 19–26. Morgan Kaufmann. 60

Blum, A. & T. M. Mitchell (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory — COLT-98*, New York, USA, pp. 92–100. ACM Press. 50, 52, 55

Blum, A. L. & P. Langley (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence 97*(1–2), 245–271. 29

Boser, B. E., I. Guyon, & V. Vapnik (1992). A training algorithm for optimal margin classifiers. In *Proceedings of Fifth ACM Workshop on Computational Learning Theory*, Pittsburgh, USA, pp. 144–152. ACM Press. 60

Boswell, T. (1990). Manual for NewId version 4.1. Technical Report TI/P2154/RAB/4/2.3, The Turing Institute. 72

Bradley, P. S. & U. M. Fayyad (1998). Refining initial points for *k-means* clustering. In J. W. Shavlik (Ed.), *Proceedings of Fifteenth International Conference on Machine Learning — ICML-1998*, Madison, Wisconsin, USA, pp. 91–99. Morgan Kaufmann. 39

Braga, A. P., A. C. P. L. F. Carvalho, & T. B. Ludermir (2003). *Redes Neurais Artificiais* (1 ed.), Chapter 6, pp. 141–168. Volume 1 of *1* (Rezende, 2003). 14

Bruce, R. (2001). A bayesian approach to semi-supervised learning. In M. Ishizuka & A. Satter (Eds.), *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium — NLPRS-2001*, Tokyo, Japan, pp. 57–64. Springer Verlag. 50, 59

Caulkins, C. W. (2000). Aquisição de conhecimento utilizando aprendizado de máquina relacional. Dissertação de Mestrado, ICMC–USP. 6

Cheeseman, P. & J. Stutz (1990). Bayesian classification (AUTOCLASS): Theory and results advances in knowledge discovery and data mining. `http://ic.arc.nasa.gov/ic/projects/bayes-group/autoclass-c-program.html`. 12, 63

Chiara, R. (2003). Aplicação de data mining em logs de servidores web. Dissertação de Mestrado, ICMC–USP. 37

Clark, P. & R. Boswell (1989). The $\mathcal{CN}2$ induction algorithm. *Machine Learning 3*(4), 261–283. 80

Clark, P. & R. Boswell (1991). Rule induction with $\mathcal{CN}2$: Some recent improvements. In Y. Kodratoff (Ed.), *Procedings of the Fifth European Working Session on Learning — EWSL-91*, pp. 151–163. Springer Verlag. `http://www.cs.utexas.edu/users/pclark/papers/newcn.ps`. 72, 80

Clark, P. & T. Niblett (1987). Induction in noise domains. In I. Bratko & N. Lavrač (Eds.), *Proceedings of the Second European Working Session on Learning — EWSL-87*, Wilmslow, UK, pp. 11–30. Sigma. 80

Collins, M. & Y. Singer (1999). Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora — EMNLP/VLC-99*, University of Maryland, MD, USA. 60

d'Alché Buc, F., Y. Grandvalet, & C. Ambroise (2002). Semi-supervised marginboost. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems 14*, Cambridge, MA, pp. 553–560. MIT Press. 59

de Sa, V. R. & D. H. Ballard (1997). Category learning through multi-modality sensing. In *Neural Computation 10(5)*, pp. 1097–1117. 56

de Sousa, E. P. M. (2003). Classificação e detecção de agrupamentos usando estruturas de indexação. Exame de Qualificação de Doutorado, ICMC–USP. 106

Decker, K. M. & S. Focardi (1995). Technology overview: A report on data mining. Technical Report CSCS TR-95-02, CSCS-ETH, Swiss Scientific Computing Center. 24

Demiriz, A. & K. Bennett (2000). Optimization approaches to semi-supervised learning. In M. Ferris, O. Mangasarian, & J. Pang (Eds.), *Complementarity: Applications, Algorithms and Extensions*, Boston, pp. 121–141. Kluwer Academic Publishers. 60

Dosualdo, D. G. (2003). Investigação de regressão para data mining. Dissertação de Mestrado, ICMC–USP. 6

Everitt, B. (1980). *Cluster Analysis*. New York: Academic Press. 28, 36

Fayyad, U. M., G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy (1996). *Advances in Knowledge Discovery and Data Mining*. California: AAAI Press / The MIT Press. 2, 24

Félix, L. C. M. (1998). Data mining no processo de extração de conhecimento de bases de dados. Dissertação de Mestrado, ICMC–USP. 6

Flach, P. A. & N. Lavrač (2000). The role of feature construction in inductive rule learning. In L. D. Raedt & S. Kramer (Eds.), *Proceedings of the Seventeenth International Conference on Machine Learning — ICML-2000 — Workshop on Attribute-Value and Relational Learning: crossing the boundaries*, Stanford, USA, pp. 1–11. Morgan Kaufmann. 30

Fung, G. & O. Mangasarian (1999). Semi-supervised support vector machines for unlabeled data classification. Technical Report 99-05, Data Mining Institute. 60

Geromini, M. R. (2002). Projeto e desenvolvimento da interface gráfica do sistema DISCOVER. Exame de Qualificação de Mestrado, ICMC–USP. 5

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley. 14

Goldman, S. & Y. Zhou (2000). Enhancing supervised learning with unlabeled data. In *Proceedings of the Seventeenth International Conference on Machine Learning — ICML-2000*, Stanford, USA, pp. 327–334. Morgan Kaufmann. 52, 56

Gomes, A. K. (2002). Análise do conhecimento extraído de classificadores simbólicos utilizando medidas de avaliação e de interessabilidade. Dissertação de Mestrado, ICMC–USP. http://www.teses.usp.br/teses/disponiveis/55/55134/tde-04072002-144610. 6

Guha, S., R. Rastogi, & K. Shim (1998). Cure: An efficient clustering algorithm for large databases. In *Proceedings ACM SIGMOD International Conference on Management of Data*, Seatle, Washington, USA, pp. 73–84. ACM Press. 3

Halembeck, G. C. (2001). Sistemas para *Clustering*. Comunicação pessoal. 46

Hamerly, G. & C. Elkan (2002). Alternatives to the *k-means* algorithm that find better clusterings. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management — CIKM-2002*, McLean, Virginia, USA, pp. 600–607. ACM Press. 39

Horst, P. S. (1999). Avaliação do conhecimento adquirido por algoritmos de aprendizado de máquina utilizando exemplos. Dissertação de Mestrado, ICMC–USP. 6

Huang, Z. (1997). A fast clustering algorithm to cluster very large categorical data sets in data mining. In *Proceedings of SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery — SIGMOD-DMKD'97*, Tucson, Arizona. 3

Imamura, C. Y. (2001). Pré-processamento para extração de conhecimento de bases textuais. Dissertação de Mestrado, ICMC–USP. 6

Ivanov, Y. A., B. Blumberg, & A. Pentland (2001). Expectation maximization for weakly labeled data. In *Proceedings of Eighteenth International Conference on Machine Learning — ICML-2001*, Williamstown, MA, USA, pp. 218–225. Morgan Kaufmann. 60

Jain, A. K. & R. C. Dubes (1988). *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall Inc. 28, 29

Jain, A. K., M. N. Murty, & P. J. Flynn (1999). Data clustering: A review. *ACM Computing Surveys 31*(3), 264–323. 3, 29, 38

John, G., R. Kohavi, & K. Pfleger (1994). Irrelevant features and the subset selection problem. In *Proceedings of the Tenth International Conference on Machine Learning — ICML-1994*, San Francisco, CA, USA, pp. 167–173. Morgan Kaufmann. 29

Kremer, S. C. & D. A. Stacey (2001). Competition: Unlabeled data for supervised learning. http://www-2.cs.cmu.edu/Groups/NIPS/NIPS2001/nips2001.html. 60

Lee, H. D. (2000). Seleção e construção de features relevantes para o aprendizado de máquina. Dissertação de Mestrado, ICMC–USP. http://www.teses.usp.br/teses/disponiveis/55/55134/tde-15032002-113112. 6, 30

Lee, H. D. & M. C. Monard (2000). Applying knowledge-driven constructive induction: Some experimental results. Technical Report 101, ICMC–USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_101.ps.zip. 30

Lee, H. D., M. C. Monard, & J. A. Baranauskas (1999). Empirical comparison of wrapper and filter approaches for feature subset selection. Technical Report 94, ICMC–USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_94.ps.zip. 29

Macqueen, J. (1967). Some methods for classification and analysis of multivariate observation. In *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Volume 1, Berkeley, pp. 281–297. University of California Press. 4, 38

Martins, C. A. (2003). Pré-processamento em mineração de textos. Tese de Doutorado, ICMC–USP. A ser defendida. 6

Martins, C. A. & M. C. Monard (2000). Interpretação de clusters utilizando aprendizado de máquina simbólico. In L. E. Vaz (Ed.), *21 Iberian Latin American Congress on Computational Methods in Engineering — CILAMCE*, Rio de Janeiro (RJ). LMC/EPUSP. pp. 13 (CD-ROM). 45

Martins, C. A., M. C. Monard, A. S. Haedo, & N. L. Matsudo (2001). Uma metodologia para auxiliar o processo de interpretação semi-automática de clusters. Technical Report 133, ICMC–USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_133.ps.zip. 45

Martins, C. A., M. C. Monard, & G. C. Halembeck (2002a). Combining clustering and inductive learning to find and interpret patterns from datasets. In *International*

Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications — CSITeA'02, Foz do Iguaçú, Brazil, pp. 90–95. 45

Martins, C. A., M. C. Monard, & G. C. Halembeck (2002b). A computational framework for interpreting clusters through inductive learning. Technical Report 173, ICMC–USP. ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_173.pdf. 45

Matsubara, E. T. (2003). Melhorando classificadores a partir de poucos exemplos rotulados. Exame de Qualificação de Mestrado, ICMC–USP. 106

Matsubara, E. T., C. A. Martins, & M. C. Monard (2002). Uma descrição dos arquivos e parâmetros utilizados na execução do sistema AUTOCLASS. Technical Report 170, ICMC–USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_170.ps.zip. 63

McCallum, A., K. Nigam, & L. H. Ungar (2000). Efficient clustering of high-dimensional data sets with application to reference matching. In R. Ramakrishnan, S. Stolfo, R. Bayardo, & I. Parsa (Eds.), *Proceedinmgs of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining — KDD-00*, New York, USA, pp. 169–178. ACM Press. 3, 24

McLachlan, G. J. & T. Krishnan (1997). *The* EM *Algorithm and Extensions*. John Wiley & Sons. 42

Melanda, E. (2002). Pós-processamento de conhecimento de regras de associação. Exame de Qualificação de Doutorado, ICMC–USP. 6

Merz, C. J. & P. M. Murphy (1998). UCI repository of machine learning datasets. University of California, Irvine, CA, USA. http://www.ics.uci.edu/~mlearn/MLRepository.html. 76, 97

Michalski, R. S., J. G. Carbonell, & T. M. Mitchell (1983). *Machine Learning: An Artificial Intelligence Approach*. Tioga Publishing Company. 13

Michalski, R. S., I. Mozetic, J. Hong, & N. Lavrac (1986). The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Fifth Annual National Conference on Artificial Intelligence*, pp. 1041–1045. 13

Michalski, R. S. & R. E. Stepp (1992). Clustering. *Encyclopedia of Artificial Intelligence*, 168–176. 3, 38

Milaré, C. R. (2003). Extração de conhecimento de redes neurais. Tese de Doutorado, ICMC–USP. 6

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence 18*(2), 203–226. Reprinted in Shavlik and Dietterich (eds.), 1990. Readings in Machine Learning, Morgan Kaufmann. 20

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Series in Computer Science. 1, 9, 58

Mitchell, T. M. (1999). The role of unlabeled data in supervised learning. In *Proceedings of the Sixth International Colloquium on Cognitive Science*, San Sebastian, Spain. (invited paper). 52, 54, 56, 57

Monard, M. C. & J. A. Baranauskas (2003). *Conceitos sobre Aprendizado de Máquina* (1 ed.), Chapter 4, pp. 89–114. Volume 1 of *1* (Rezende, 2003). 23

Nagai, W. A. (2000). Ambiente de avaliação de conhecimento para problemas de regressão. Dissertação de Mestrado, ICMC–USP. 6

Nigam, K. & R. Ghani (2000). Analyzing the effectiveness and applicability of Co-Training. In *Proceedings of the Ninth International Conference on Information and Knowledge Management — CIKM-2000*, McLean, Virginia, USA, pp. 86–93. ACM Press. 57

Nigam, K., A. K. McCallum, S. Thrun, & T. M. Mitchell (2000). Text classification from labeled and unlabeled documents using EM . *Machine Learning 39*(2/3), 103–134. 59

Paula, M. F. (2003). Ambiente para disponibilização de conhecimento. Dissertaçao de Mestrado, ICMC–USP. 6

Pila, A. D. (2001). Seleção de de atributos relevantes para aprendizado de máquina utilizando a abordagem de rough sets. Dissertação de Mestrado, ICMC–USP. http://www.teses.usp.br/teses/disponiveis/55/55134/tde-13022002-153921/publico/dissertacao_ADP.pdf. 6

Prati, R. C. (2003). O *framework* de integração do sistema Discover. Dissertação de Mestrado, ICMC–USP. 5

Prati, R. C., J. A. Baranauskas, & M. C. Monard (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. 6

Pugliesi, J. B. (2001). O pós-processamento em extração de conhecimento de bases de dados. Exame de Qualificação de Doutorado, ICMC–USP. 6

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning 1*, Volume 1, pp. 81–106. Kluwer Academic Publishers. 19, 72

Quinlan, J. R. (1987). Generating production rules from decision trees. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, pp. 304–307. Morgan Kaufmann. 13, 72

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Los Altos, California, USA: Morgan Kaufmann. 72, 80

Raggett, J. & W. Bains (1992). *Artificial Intelligence from A to Z*. Chapman & Hall. 10

Rezende, S. O. (2003). *Sistemas Inteligentes: Fundamentos e Aplicações*. Barueri, SP, Brasil: Editora Manole. 113, 118, 119

Rezende, S. O., J. B. Pugliesi, E. A. Melanda, & M. F. Paula (2003). *Mineração de Dados* (1 ed.), Chapter 12, pp. 307–336. Volume 1 of *1* (Rezende, 2003). 2

Riloff, E. & R. Jones (1999). Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, Orlando, Florida, USA, pp. 474–479. AAAI Press. 56

Rozante, T. A. A. (2003). Implantação do reuso de componentes no processo de desenvolvimento de software. Dissertação de Mestrado, ICMC–USP. 5

Russel, S. & P. Norvig (2003). *Artificial Intelligence: A Modern Approach* (2 ed.). Prentice Hall. 21

Sanches, M. K. & M. C. Monard (2003). Avaliação experimental de um algoritmo de aprendizado de máquina semi-supervisionado para auxiliar na rotulação de exemplos. Technical report, ICMC–USP. Em Andamento. 96

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 197–227. 59

Seeger, M. (2002). Covariance kernels from bayesian generative models. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems — NIPS*, Volume 14, Cambridge, MA. MIT Press. 60

Simon, H. E. (1983). Search and reasoning in problem solving. *Artificial Intelligence 21*, 7–29. 10

Sneath, P. H. A. & R. R. Sokal (1973). *Numerical Taxonomy*. San Francisco, USA: Freeman Press. 36

Stanfill, C. & D. Waltz (1986). Instance-based learning algorithms. *Communications of the ACM 12*, 1213–1228. 105

Stepp III, R. E. & S. Michalski (1986). Conceptual Clustering: *Inventing Goal-Oriented Classifictions of Structured Objects*, Volume 2, Chapter 17, pp. 471–478. Morgan Kaufmann. 27

Wagstaff, K., C. Cardie, S. Rogers, & S. Schroedl (2001). Constrained *k-means* clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning — ICML-2001*, Williamstown, MA, USA, pp. 577–584. 50

Wall, L., T. Christiansen, & R. L. Schwartz (1996). *Programming Perl* (2 ed.). O'Reilly & Associates. 65, 70

Witten, I. H. & E. Frank (2000). *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations*, Volume 1. Morgan Kaufmann. http://www.cs.waikato.ac.nz/ml/weka/index.html. 37, 41, 43