Um Método Geral para Tornar Algoritmos Fuzzy de Aprendizado de Máquinas Escaláveis para Bases de Dados Arbitrariamente Grandes

por

Thiago Petinari Silva Cordeiro

UFRJ

Tese submetida para a obtenção do título de

Mestre em Ciências em Engenharia de Sistemas e Computação

ao Programa de Pós-Graduação de Engenharia de Sistemas e Computação da COPPE/UFRJ

por

Thiago Petinari Silva Cordeiro

Junho 2006

UM MÉTODO GERAL PARA TORNAR ALGORITMOS FUZZY DE APRENDIZADO DE MÁQUINAS ESCALÁVEIS PARA BASES DE DADOS ARBITRARIAMENTE GRANDES

Thiago Petinari Silva Cordeiro

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Gerson Zaverucha, Ph.D.

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. Teresa Bernarda Ludermir, Ph.D.

RIO DE JANEIRO, RJ - BRASIL
JUNHO DE 2006

CORDEIRO, THIAGO PETINARI SILVA

Um Método Geral para Tornar Algoritmos Fuzzy de Aprendizado de Máquinas Escaláveis para Bases de Dados Arbitrariamente Grandes [Rio de Janeiro] 2006

XV, 85 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2006)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1 - Escalabilidade de algoritmos

2 - Amostragem

3 - Agrupamento fuzzy

4 - Mineração de grandes bases de dados

I. COPPE/UFRJ II. Título (série)

Agradecimentos

Agradeço primeiramente a Deus, o único que esteve sempre comigo, mesmo nas noites longas em claro.

Agradeço aos meus pais, que me deram a vida, me educaram, me transmitiram sabedoria e me apoiaram sempre, em todas as minhas escolhas até aqui, e sei que continuarão me apoiando, na melhor das alegrias ou na pior das tristezas.

Agradeço à minha família que me fez sentir em casa, tanto no Brasil quanto em Portugal.

Agradeço ao meu professor e orientador Gerson Zaverucha, pela compreensão nas dificuldades, pela orientação e por ter sido o meu principal guia, sem o qual este trabalho não existiria.

Agradeço ao professor Pedro Domingos por ter sido sempre solícito quando precisei.

Agradeço a CAPES pelo suporte financeiro.

Agradeço aos meus anjos da guarda que tive durante esses anos, que me incentivaram a ultrapassar os obstáculos e me inspiraram a crescer e conseguir meus objetivos.

Agradeço aos amigos que ganhei e aos amigos que perdi, com a certeza de saber que todos eles ajudaram a formar quem sou hoje.

Agradeço a Roberto Gomes Bolaños, por ter me acompanhado desde a infância e por seu talento, que até hoje é minha válvula de escape em todos os momentos.

Por fim, agradeço a Peter Parker, em quem sempre me inspirei e que sempre vai ser um herói e um espelho para mim.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UM MÉTODO GERAL PARA TORNAR ALGORITMOS FUZZY DE APRENDIZADO DE MÁQUINAS ESCALÁVEIS PARA BASES DE DADOS ARBITRARIAMENTE GRANDES

Thiago Petinari Silva Cordeiro

Junho/2006

Orientador: Gerson Zaverucha

Programa: Engenharia de Sistemas e Computação

Domingos e Hulten desenvolveram uma metodologia genérica para tornar algoritmos de aprendizado de máquina escaláveis e aplicaram essa metodologia ao algoritmo K-Means. O objetivo deste trabalho é adaptá-lo para tornar algoritmos de aprendizado de máquina fuzzy escaláveis para bases de dados arbitrariamente grandes. Como cada exemplo de um algoritmo de aprendizado fuzzy está associado com cada classe/*cluster* através da matriz de pertinência, nós tivemos que alterar todo cálculo do erro do aprendizado usando nossas definições de exemplos *fuzzy sampling false positives* e *fuzzy sampling false negatives*. Então, nós aplicamos esse método para o Fuzzy C-Means (FCM), desenvolvendo o Very Fast Fuzzy C-Means (VFFCM). De forma similar ao Very Fast K-Means (VFKM) de Domingos e Hulten, VFFCM utiliza menos exemplos (determinado pelo teoricamente limite de Hoeffding) a cada passo garantindo que o modelo resultante não difira significantemente daquele que seria produzido passando todos os dados pelo FCM. VFFCM é comparado com o FCM e o VFKM, demonstrando, respectivamente, seu *speedup* e melhor qualidade de agrupamento (usando para comparação os verdadeiros *clusters* da base de dados).

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A METHOD FOR SCALING UP FUZZY MACHINE LEARNING ALGORITHMS TO ARBITRARILY LARGE DATABASES

Thiago Petinari Silva Cordeiro

June/2006

Advisor: Gerson Zaverucha

Department: Systems Engineering and Computer Science

Domingos and Hulten developed a general framework to scale up machine learning algorithms and applied it to K-Means. The objetive of this work is adapt it to scale up fuzzy algorithms to arbitrarily large databases. Since each example in fuzzy learners is associated with every class/cluster through the membership matrix, we have to change the entire calculation of the learner's error using our definitions of fuzzy sampling false positives and fuzzy sampling false negatives examples. Then, we apply this method to Fuzzy C-Means (FCM), developing the Very Fast Fuzzy C-Means (VFFCM). Similarly to Domingos and Hulten Very Fast K-Means (VFKM), VFFCM uses less examples (determined theoretically by Hoeffding bound) in each step still guaranteeing that the resulting model does not significantly differ from the one that would be created passing the entire data through the FCM. VFFCM is compared to FCM and VFKM showing its speedup and better quality clustering (using the true database clusters), respectively.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

A Algoritmo

AFCM *Accelerated/Alternative/Approximate Fuzzy C-Means*

ARFLC *Adaptive Rough Fuzzy Leader Clustering*

BCFCM *Bias-Correct Fuzzy C-Means*

BD Banco de Dados

c *Cluster*

C/S Índice de Compactação/Separação

CFCM *Conditional Fuzzy C-Means*

CVFDT *Concept-adapting Very Fast Decision Tree*

D Dimensionalidade do problema

eFFCM *extensible Fast Fuzzy C-Means*

EM *Expectation-Maximization*

FCM *Fuzzy C-Means*

FCMED *Fuzzy C-Medians*

FCMP *Fuzzy C-Means with Feature Partitions*

FFCM *Fast Fuzzy C-Means*

FJM *Fuzzy J-Means*

FKM *Fuzzy K-Means*

FKME *Fuzzy K-Means with Extragrades*

FSCM *Fuzzy symbolic c-Means*

fsfn *Fuzzy Sampling False Negatives*

fsfp *Fuzzy Sampling False Positives*

GB *Gigabytes*

GHz *Gigahertz*

geFFCM *generalized extensible Fast Fuzzy C-Means*

H Matriz de distância do FCM

H' Matriz de distância do VFFCM

HCM *Hard C-Means*

HUFC *Hierarchical Unsupervised Fuzzy Clustering*

I Número máximo de iterações

K Número de clusters

KDD *Knowledge Discovery in Databases*

KM *K-Means*

L(...) Função de perda

LFCM *Literal Fuzzy C-Means*

m Parâmetro de nebulosidade

M Modelo produzido

MB *Megabytes*

MS-FCM *Modified Suppressed Fuzzy C-Means*

N Tamanho máximo da base de dados

PAC *Probably Approximately Correct*

PBM *Pakhira Bandyopadhyay Maulik index*

PBMF *Fuzzy PBM index*

PC *Partition Coefficient index*

PS *Progressive Sampling*

R Domínio das variáveis

RAM *Random Access Memory*

S Matriz de subtração

S-FCM *Suppressed Fuzzy C-Means*

T Conjunto de exemplos

U Matriz de pertinência do FCM

U' Matriz de pertinência do VFFCM

VFBN *Very Fast Bayesian Network*

VFDT *Very Fast Decision Tree*

VFEM *Very Fast Expectation-Maximization*

VFFCM *Very Fast Fuzzy C-Means*

VFKM *Very Fast K-Means*

VFREL *Very Fast Relational Learning*

XB *Xie-Beni index*

Capítulo 1

Introdução

Neste capítulo, o problema abordado pelo trabalho será apresentado, bem como a solução desse problema, que será o objetivo desta dissertação. A organização dos capítulos desse trabalho também será brevemente apresentada.

1.1 Descrição do Problema

Devido à crescente quantidade de dados disponíveis hoje em dia, é cada vez maior, também, a necessidade de desenvolver novas técnicas e ferramentas capazes de processar e analisar toda essa enorme quantidade de dados, de forma inteligente, automática e eficiente, descobrindo informações úteis e valiosas. O campo de pesquisa responsável pelo desenvolvimento dessas técnicas e ferramentas é chamado de Extração de Conhecimento de Bases de Dados (*Knowledge Discovery in Databases - KDD*) (FRAWLEY, et al., 1991; PIATETSKY-SHAPIRO, FRAWLEY, 1991; FAYYAD, et al., 1996b). KDD é o processo de identificar padrões ou modelos que representem informação válida, inédita, potencialmente útil e essencialmente compreensível em uma coleção de dados (FAYYAD, 1998).

Uma das áreas mais pesquisadas no campo da Mineiração de Dados (*Data Mining*) e KDD é o agrupamento (*clustering*), cujos algoritmos permite organizar os exemplos similares do banco de dados em diversos grupos (*clusters*). Porém, o tamanho das bases de dados disponíveis chegaram a um patamar muito difícil de ser agrupado com os algoritmos comuns existentes na literatura: simples e de fácil compreensão e implementação, porém custosos em tempo de execução.

Assim, hoje em dia, um dos maiores desafios da comunidade de Mineração de Dados e KDD tem sido tornar os algoritmos de Aprendizado de Máquina (*Machine Learning*) (MITCHELL, 1997) escaláveis para grandes bancos de dados. Uma grande quantidade de artigos tem focado nesse tema (NG, HAN, 1994; ZHANG, et al., 1996; ESTER, et al., 1996; GANTI, et al., 1999; PROVOST, KOLLURI, 1999; PROVOST, et al., 1999; FARNSTROM, et al., 2000; BEZDEK, HATHAWAY, 2004).

Uma pequena descrição de cada um desses trabalhos:

- (NG, HAN, 1994) apresenta o método de agrupamento CLARANS, baseado em busca aleatória, além de dois novos algoritmos que utilizam esse método para mineração de bancos de dados espaciais, que costumam ser enormes.

- (ZHANG, et al., 1996) apresenta o método denominado BIRCH, capaz de obter um bom agrupamento com uma única passagem pela base de dados, e ir melhorando sua qualidade através de passagens adicionais. Em seus experimentos o BIRCH mostrou-se consistentemente superior ao método CLARANS.

- (ESTER, et al., 1996) desenvolveu o algoritmo DBSCAN, também para bases de dados espaciais, inclusive em dados com ruídos. Foi mostrado

que o DBSCAN superou o método CLARANS em um fator maior que 100 em termos de eficiência.

- (GANTI, et al., 1999) apresenta um resumo de métodos de escalonamento para três problemas clássicos da mineração de dados: análise de associações (*market basket analysis*), agrupamento e classificação.

- (PROVOST, KOLLURI, 1999) também apresenta um resumo de métodos de escalonamento, porém mais amplo, categorizando e comparando os mais diversos métodos existentes.

- (PROVOST, et al., 1999) desenvolveu o método de escalonamento chamado amostragem progressiva (*progressive sampling*), que utiliza subamostras do banco de dados para melhorar a eficiência dos algoritmos. Em outro capítulo desta dissertação, mais detalhes serão dados sobre esse método.

- (FARNSTROM, et al., 2000) apresenta um algoritmo que executa o algoritmo de agrupamento K-Means através de uma única passagem pela base de dados (Single Pass K-Means) e mostra através de experimentos que os resultados deste algoritmo são iguais ou quase iguais aos do K-Means original.

- (BEZDEK, HATHAWAY, 2004) aplica o método de amostragem progressiva (PROVOST, et al., 1999) no algoritmo eFFCM (PAL, BEZDEK, 2002), originalmente desenvolvido para agrupamento de grandes imagens digitais, criando o geFFCM, aplicável a qualquer base de dados.

- etc.

A definição de *escalabilidade* é: um algoritmo é *escalável* se sua complexidade de tempo de execução aumenta linearmente com o número de exemplos dos dados de entrada (GANTI, et al., 1999). O problema central de escalabilidade não é aumentar a velocidade de um algoritmo lento e sim transformar um algoritmo impraticável em um que possa ser aplicado de forma eficiente. Portanto, não é o "quão rápido" pode-se executar um problema e sim qual a dimensão do problema que pode-se lidar. Do ponto de vista de análise de complexidade, para a maior parte dos problemas de escalabilidade o fator limitante é o número de exemplos do banco de dados. Um número grande de exemplos introduz problemas em potencial na complexidade de tempo e espaço. Um milhão de exemplos (entre 100MB e 1GB) é considerado como uma base de dados muito grande (HUBER, 1996).

Os três métodos principais de escalabilidade, apresentados na Tabela 1.1, são: desenvolvimento de algoritmos rápidos, particionamento de dados e representação relacional. Esta dissertação será focada nos dois últimos métodos. Técnicas de particionamento de dados, como podemos ver na Tabela 1.2 são caracterizadas de duas formas: pela separação de subconjuntos de exemplos ou de subconjuntos de atributos. A primeira técnica, que passaremos a chamar de agora em diante de amostragem (*sampling*), é a abordagem mais utilizada para lidar com a intratabilidade de aprender a partir de grandes conjuntos de dados e consiste em selecionar uma única amostra (subconjunto) de exemplos, de tamanho menor que o tamanho total do banco de dados, para representação do mesmo. Nesta pesquisa focaremos nessa abordagem.

A amostragem é uma técnica bem aceita na comunidade de Estatística: "...um processo poderoso, computacionalmente intenso, operando em um subconjunto dos dados de fato, pode prover uma acurácia superior do que um processo menos sofisticado usando toda base de dados." (FRIEDMAN,

Abordagem Principal	**Método Geral**
Desenvolvimento de Algoritmos Rápidos	Espaço de modelos restrito
	Heurísticas de busca poderosas
	Otimização de algoritmo/programação
	Paralelização
Particionamento de Dados	Selecionar um subconjunto de exemplos
	Selecionar um subconjunto de atributos
	Processar subconjuntos de forma seqüencial
	Processar subconjuntos de forma concorrente
Representação Relacional	Representar dados de forma relacional
	Integrar mineração de dados com gerenciamento da base de dados

Tabela 1.1: Métodos de Escalabilidade

Método Geral	**Exemplo de Técnica**
Selecionar um subconjunto de exemplos (Amostragem)	Amostragem randômica
	Compactação dupla
	Amostragem por extração
	Amostragem por megaindução (*Peepholing*)
Selecionar um subconjunto de atributos	Uso de conhecimento relevante
	Uso de indicadores estatísticos
	Uso de estudos de subconjuntos
Processar subconjuntos de forma seqüencial	Aprendizado de multi-subconjuntos independente
	Aprendizado de multi-subconjuntos seqüencial
Processar subconjuntos de forma concorrente	Aprender modelos múltiplos, utlizar o melhor
	Combinar descrições de classe
	Combinar predições
	Aprendizado cooperativo

Tabela 1.2: Exemplos de Particionamento de Dados

1997). Uma questão central é determinar o tamanho da amostra a ser utilizado, pois depende de fatores desconhecidos a priori. Uma abordagem é dada pela complexidade da amostra, no ambiente de aprendizado teórico PAC (*Probably Approximately Correct*).

PAC é um modelo de aprendizado introduzido por (VALIANT, 1984) que afirma que o aprendizado de uma função para uma classe desconhecida pode ser vinculada, com alta probabilidade, à obtenção de uma hipótese que é uma boa aproximação desta função. O modelo PAC foi aprimorado (HAUSSLER, 1988), generalizado (HAUSSLER, 1990) e em (HAUSSLER, 1992) foi aplicado à vários algoritmos de aprendizado, principalmente redes neurais. Posteriormente, (MISHRA, et al., 2001) aplicou o PAC em algoritmos de agrupamento.

No entanto, a abordagem PAC fornece limites que tornam muitas vezes seu uso impraticável, já que exige uma amostra muito grande, o que não aumentaria a eficiência do aprendizado. Isso acontece porque são aplicados a união dos limites a todos os modelos da classe; o número de modelos geralmente é grande e, portanto, quanto mais modelos, pior é o limite. Esse é um problema conhecido no ambiente PAC.

1.2 Objetivo do Trabalho

Em (DOMINGOS, HULTEN, 2001; DOMINGOS, HULTEN, 2003), um método geral para tornar algoritmos de aprendizado de máquinas escaláveis foi desenvolvido e aplicado a diferentes tipos de algoritmos: árvores de decisão (VFDT (DOMINGOS, HULTEN, 2000), CVFDT (HULTEN, et al., 2001)), redes Bayesianas (VFBN (HULTEN, DOMINGOS, 2002)), EM (VFEM (DOMINGOS, HULTEN, 2002)), modelos relacionais (VFREL (HULTEN, et al.,

2003)), e quando foi aplicado ao algoritmo de agrupamento K-Means, o algoritmo Very Fast K-Means (VFKM (DOMINGOS, HULTEN, 2001)) foi desenvolvido.

O método cria um novo algoritmo que utiliza um limite máximo para a perda do aprendizado, em função do número de exemplos, e usa esse limite para reduzir esse número de exemplos a cada passo do algoritmo (determinado teoricamente pelo limite de Hoeffding (HOEFFDING, 1963)), garantindo que o modelo resultante não difira significantemente do modelo que seria produzido passando todos os dados pelo algoritmo original. Esse limite requer uma amostra bem menor que no PAC, tornando a metodologia prática.

Apesar do método poder ser usado em outros algoritmos de aprendizado de máquinas *crisp* (*hard*), ele não é aplicável à algoritmos nebulosos (*fuzzy* ou *soft*). Como cada exemplo de um algoritmo fuzzy é associado a cada classe/cluster pela matriz de pertinência, para usar essa metodologia nós tivemos que alterar todo o cálculo de erro do algoritmo, usando nossas definições de exemplos *fuzzy sampling false positives* e *fuzzy sampling false negatives*.

Fuzzy C-Means (também conhecido como Fuzzy K-Means, FKM ou como iremos chamá-lo de agora em diante, FCM) é uma versão fuzzy do K-Means. Escolhemos o FCM pois quando comparado ao K-Means, vários trabalhos verificaram vantagens em sua aplicação em termos de eficiência e desempenho (BEZDEK, et al., 1984; CANNON, et al., 1986; BEZDEK, et al., 1999; HAMERLY, ELKAN, 2002). Portanto, para exemplificar o método criado, nós o aplicamos ao FCM, desenvolvendo um novo algoritmo, o Very Fast Fuzzy C-Means (VFFCM), que reduz o número de exemplos usados a cada passo.

Ele tem a garantia que o modelo produzido não difira significantemente com o modelo que seria criado pelo FCM.

Depois do algoritmo criado, o implementamos e aplicamos a um conjunto de 30 bancos de dados (*datasets*), gerados artificialmente, para verificar sua melhora de agrupamento e eficiência em relação ao FCM e ao VFKM. Aproveitamos a oportunidade para também fazer uma comparação entre três índices de validação de qualidade de agrupamento: índice de coeficiente de partição, índice de compactação/separação e índice PBM(F).

Em (PAL, BEZDEK, 2002) foi aplicado um método estatístico de escalabilidade no algoritmo FCM, especificamente para solucionar problemas relacionados à processamento de imagens (*image processing*) (GONZALEZ, WINTZ, 1987; JAIN, 1989) e visão computacional (*computer vision*) (WINSTON, 1975; BROWN, 1988). Naquele trabalho houve uma pesquisa sobre aplicação de métodos estatísticos para tornar o algoritmo FCM escalável para grandes bases de dados e foi dito que "...assegurar um limite assintótico na faixa de erro para o FCM seguindo o método (DOMINGOS, HULTEN, 2001) para o C-Means crisp é uma idéia interessante e útil para uma futura pesquisa." e quatro anos depois, em (HATHAWAY, BEZDEK, 2006), ainda foi dito pelos mesmos autores que "A análise para o caso crisp não é facilmente generalizada para o caso fuzzy; esse empreendimento ambicioso ainda não foi feito." Nosso objetivo é mostrar essa análise almejada finalmente pronta (CORDEIRO, ZAVERUCHA, 2005; CORDEIRO, ZAVERUCHA, 2006).

1.3 Organização do Trabalho

No capítulo 2 os conhecimentos preliminares são brevemente apresentados: K-Means, FCM, três índices de validação de qualidade de agrupa-

mento, e a técnica de amostragem progressiva. No capítulo 3 o algoritmo VFKM (DOMINGOS, HULTEN, 2001) será mostrado, bem como suas características principais. No capítulo 4 o método é apresentado e nós definimos fuzzy sampling false positives e fuzzy sampling false negatives, dois conceitos fundamentais para a aplicação do método. No capítulo 5 a metodologia é adaptada ao algoritmo FCM para desenvolver o novo algoritmo VFFCM. No capítulo 6 os resultados experimentais são apresentados. Finalmente, no capítulo 7 nós fornecemos as conclusões obtidas neste trabalho.

Capítulo 2

Conhecimentos Preliminares

Agrupamento (*Clustering*) (KAUFMAN, ROUSSEEUW, 1990) é uma importante área do aprendizado de máquinas (MITCHELL, 1997) e pode ser aplicada a muitos campos, incluindo mineração de dados (FAYYAD, et al., 1996b), análise de dados estatísticos (KAUFMAN, ROUSSEEUW, 1990; BANFIELD, RAFTERY, 1993; FAYYAD, et al., 1996a), aplicações de negócios (BRACHMAN, et al., 1996), entre muitos outros estudos, de medicina à agricultura (EVERITT, 1974; FUKUNAGA, 1990; JAIN, DUBES, 1988; NG, HAN, 1994; ESTER, et al., 1995; ZHANG, et al., 1996; WANG, et al., 1997; FAYYAD, et al., 1998; HINNEBURG, KEIM, 1998; SHEIKHOLESLAMI, et al., 1998; AGRAWAL, et al., 1998; BRADLEY, et al., 1998; GUHA, et al., 1998; GUHA, et al., 1999; AGGARWAL, et al., 1999; WANG, et al., 1999; FARNSTROM, et al., 2000; FARNSTROM, LEWIS, 2000; JAIN, LAW, 2005).

Na comunidade de mineração de dados não existe uma definição rígida para o procedimento de agrupamento que, assim, foi formulado de várias maneiras: para aprendizado de máquinas (FISHER, 1987), reconhecimento de padrões (DUDA, HART, 1973; FUKUNAGA, 1990), otimização (SELIM,

ISMAIL, 1984; BRADLEY, et al., 1996) e estatística (SILVERMAN, 1986; KAUFMAN, ROUSSEEUW, 1990; SCOTT, 1992; BANFIELD, RAFTERY, 1993; BISHOP, 1995). De forma geral, (GUHA, et al., 1998) define agrupamento como segue: Dados N pontos em um espaço de D dimensões, os particionamos em K grupos, tal que os pontos de um grupo são mais similares entre si do que com os pontos dos demais grupos (que, de agora em diante, serão chamados apenas de *clusters*). Assim, o problema central do agrupamento é reunir (agrupar, "clusterizar") itens similares de dados. Assim, algoritmos de agrupamento procuram encontrar estruturas de dados intrínsecas, e de acordo com o critério de relacionamento adotado, organizar objetos de dados similares em clusters.

Clusters, em um determinado espaço, são definidos como regiões conectadas naquele espaço, contendo uma densidade relativamente alta de pontos, separados de outros clusters por regioes esparsas (EVERITT, 1974).

A abordagem feita pelo agrupamento é comumente chamada de aprendizado não supervisionado, pois não há classes explícitamente descritas denotando, a priori, a partição a qual o dado pertence, contrastando com o aprendizado supervisionado (classificação), onde os objetos possuem classes definidas.

Técnicas de agrupamento podem ser divididas em dois métodos (JAIN, DUBES, 1988; HAN, KAMBER, 2001): métodos hierárquicos e métodos de partição baseados em uma função objetivo. Métodos hierárquicos manipulam hierarquias completas, isto é, uma sequência aninhada de partições de dados (JAIN, et al., 1999). Métodos de partição têm por objetivo substituir a partição única do conjunto de entrada em um número fixo de grupos, minimizando uma função de custo (a função objetivo) enquanto procura atingir um critério ótimo para o modelo (DIDAY, SIMON, 1976).

Nas sub-seções a seguir nós apresentaremos dois dos principais algoritmos de agrupamento existentes: K-Means e FCM. Eles utilizam o método de partição para o agrupamento dos dados. Apresentaremos também três índices validação, para avaliação da qualidade de agrupamento dos algoritmos. Por último, será descrito neste capítulo o funcionamento do método de amostragem progressiva.

2.1 K-Means

Originalmente conhecido como método de Forgy (FORGY, 1965), o algoritmo K-Means (MACQUEEN, 1967) (também conhecido como Hard C-Means) é um dos algoritmos de agrupamento mais importantes e conhecidos na comunidade de mineração de dados. Apesar disso ele não é aplicável a bancos de dados arbitrariamente grandes, por ser um algoritmo custoso em tempo de execução.

Na Figura 2.8 o pseudo-código do K-Means é mostrado, juntamente com o algoritmo FCM, que será abordado na seção seguinte. Dado um conjunto de N exemplos $\{x_1, ..., x_N\}$ que precisam ser agrupados, o algoritmo K-Means funciona da seguinte forma. Inicialmente, K clusters, representados pelos seus respectivos centros que chamaremos de *centroids*, são definidos randomicamente ou através de alguma heurística (MEILA, HECKERMAN, 1998; PENA, et al., 1999). A cada iteração, cada exemplo será associado ao cluster mais próximo. Depois de todos os exemplos terem sido "ganhos" por algum cluster, cada centroid é recalculado computando a média dos exemplos associados ("ganhos") a ele. Esses passos se repetem até que os centroids não se movam mais, de acordo com um limiar γ especificado inicialmente, mostrando assim que o algoritmo alcançou um mínimo (local ou global). A saída

do K-Means é um conjunto de K centroids, cada um deles sendo o ponto central que representa o seu respectivo cluster.

Existem algumas condições de término normalmente utilizadas na literatura. Uma das mais conhecidas, que usamos ao longo do trabalho, é a soma das diferenças de todas as variações dos clusters ao quadrado. Essa soma irá decair até alcançar o limiar γ especificado.

A função de distância mais difundida na literatura é a métrica Minkowski, definida como: $dist(a, b) = \sqrt[e]{\sum_d |a_d - b_d|^e}$ onde a e b são dois pontos e d é a dimensionalidade desses pontos. O parâmetro e especifica a métrica utilizada, isto é, métrica Chebychev ($e = \infty$), métrica Manhattan ($e = 1$) ou métrica Euclideana ($e = 2$). A escolha da métrica a ser usada depende do tipo de problema a ser abordado e sua mudança pode alterar substancialmente os resultados obtidos (WU, YANG, 2002). Foi decido por usar a métrica Euclideana ao longo do trabalho e nos experimentos.

Na Figura 2.1 temos um exemplo gráfico de um banco de dados simples, de duas dimensões (atributos), em que cada um dos seus N exemplos é representado por um ponto. É visualmente clara a existência de 5 grupos distintos de pontos, assim, utilizando $K = 5$, um bom agrupamento dessa banco está representado na Figura 2.2, onde a cada área representa um cluster na divisão fornecida pelo modelo de agrupamento produzido pelo algoritmo. Os pontos maiores representam os centroids de cada cluster.

Porém, como já foi dito, o algoritmo K-Means está sujeito a cair em mínimos locais. Na Figura 2.3 pode ser visto outro possível agrupamento dos exemplos, onde o conjunto de pontos mais à esquerda acabou por ser dividindo entre os dois clusters C_3 e C_4, enquanto um único cluster C_5 "ganhou" dois conjuntos de pontos, que faziam parte de clusters diferentes no agrupamento anterior (Figura 2.2). Essa, evidentemente, não foi um agrupamento

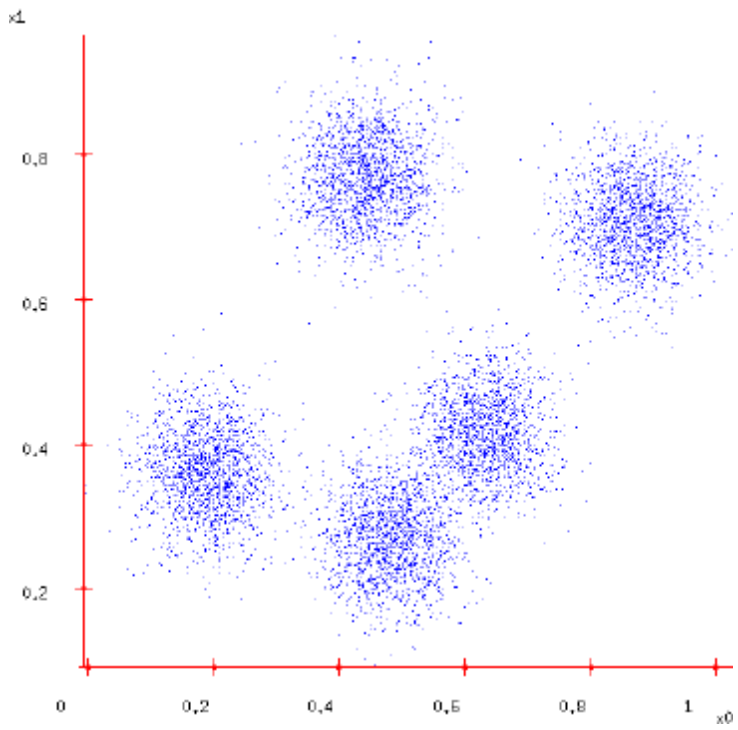

Figura 2.1: Exemplo de Banco de Dados

tão bom, pois não representou de forma satisfatória a divisão claramente existente no banco de dados.

Uma importante característica do K-Means diz respeito ao fato que o número de clusters K ser um parâmetro de entrada fixo. Vários métodos foram desenvolvidos para escolha de um K inicial (PELLEG, MOORE, 2000; BRADLEY, FAYYAD, 1998), porém como esse não é o escopo do trabalho, utilizamos apenas um K fixo, com uma escolha aleatória de inicialização através de exemplos randômicos do banco de dados.

Outra característica básica do K-Means é o seu funcionamento crisp, onde cada exemplo é sempre "ganho" por apenas um cluster.

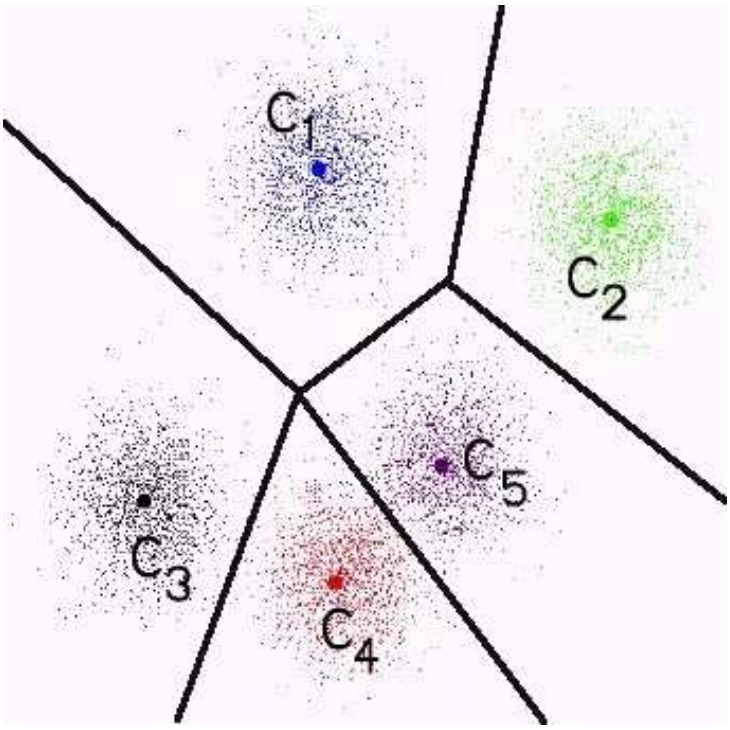

Figura 2.2: Exemplo de Agrupamento - Mínimo Global

2.2 FCM

FCM é um algoritmo de agrupamento fuzzy desenvolvido por (DUNN, 1973) e posteriormente aprimorado por (BEZDEK, 1981). Suas vantagens em relação ao K-Means, comparando a qualidade do agrupamento, foram verificadas em diversas aplicações (BEZDEK, et al., 1984; CANNON, et al., 1986; BEZDEK, et al., 1999; HAMERLY, ELKAN, 2002). Isto se verificou pois o comportamento soft é uma característica inerente a muitas aplicações existentes no mundo real.

Os conjuntos fuzzy (ZADEH, 1965; MCNEILL, FREIBERGER, 1993) foram desenvolvidos para representar as medidas de incerteza existentes nos fenômenos aleatórios do mundo real. Enquanto a lógica binaria tradicional

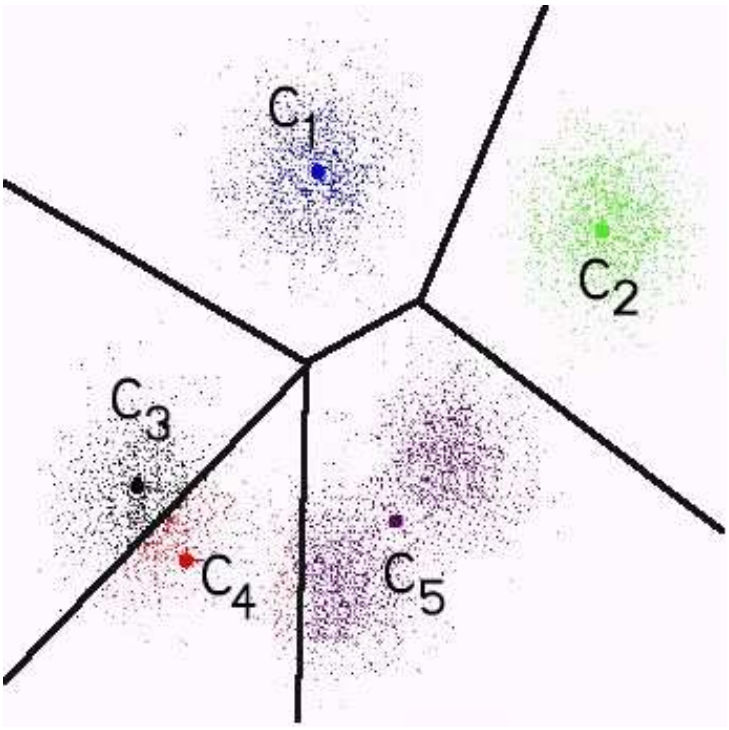

Figura 2.3: Exemplo de Agrupamento - Mínimo Local

impõe limites rígidos, a lógica fuzzy permite valores intermediários definidos entre os valores convencionais de sim/não, verdadeiro/falso, 0/1, etc.

Na Figura 2.4 temos um exemplo simples de um banco de dados em formato de borboleta e seu agrupamento no caso hard, utilizando $K = 2$ (os pontos envolvidos pela linha contínua pertencem ao cluster C_1 e pela linha tracejada fazem parte do cluster C_2). Já na Figura 2.5, temos um agrupamento fuzzy. Percebe-se neste caso que o ponto central passou a ter grau de pertinência $0,5$ relativo à cada um dos clusters, o que é natural, já que sua distância é a mesma em relação aos dois centroids. Porém como o K-Means associa cada exemplo a um único cluster, na Figura 2.4 ele teve que ser "ganho" por um dos clusters (a escolha do cluster depende da implementação,

mas geralmente é o primeiro ao qual o ponto é atribuído). O algoritmo FCM, produzindo o modelo soft da Figura 2.5, eliminou essa limitação.

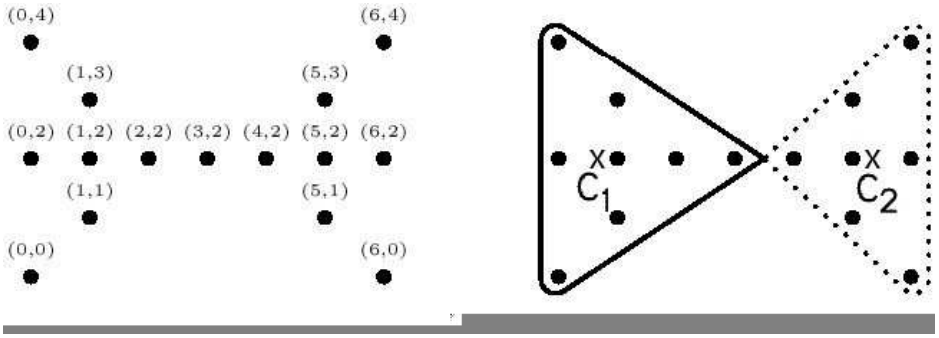

Figura 2.4: Exemplo de Banco de Dados Borboleta e seu Agrupamento Crisp

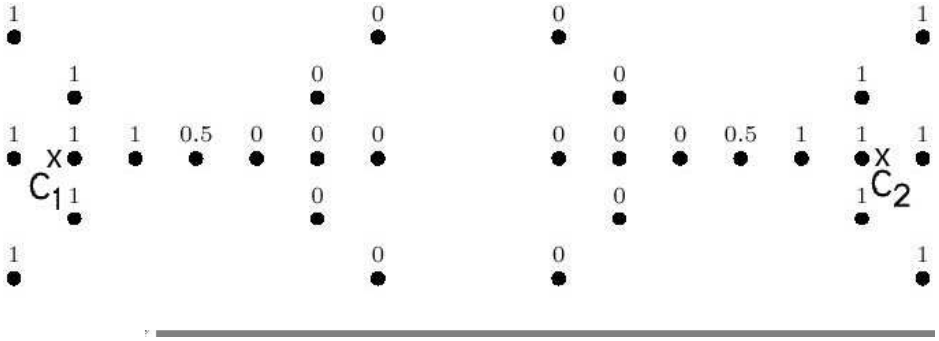

Figura 2.5: Agrupamento Fuzzy ($m = 1,25$) - À Esquerda os Graus de Pertinência relativos ao C_1 e à Direita os Graus de Pertinência relativos ao C_2

Por ser um algoritmo simples de ser estudado e implementado, o FCM tem sido aplicado com sucesso nas mais diversas áreas, tais como:

- **Biologia** - A partir da composição de amino-ácidos, prever uma das 4 classes de estrutura das proteínas (alpha, beta, alpha+beta ou alpha/beta) (ZHANG, CHOU, 1995).

- **Química** - Fazer reconhecimento de padrões em atividade molecular, onde a partir do algoritmo GFCA, porém utilizando o critério de perturbação de forma diferente, foi possível criar o novo algoritmo FCASA (SUN, DANZER, 1996) ou identificar componentes químicos para química combinatorial de forma consistente (LINUSSON, et al., 1998).

- **Meteorologia** - Classificar dados climáticos de forma fuzzy (MCBRATNEY, MOORE, 1985).

- **Gastronomia** - Qualificar produtos relacionados à piscicultura através da aplicação e aprimoramento de técnicas fuzzy (HU, et al., 1998).

- **Geologia** - Análisar a geoquímica do terreno através de sedimentos (RANTITSCH, 2000) e através de mapas em alta resolução para classificação do seu formato e geologia (BURROUGH, et al., 2000).

- **Hidroquímica** - Detectar a tipologia físico-química da água através da análise do terreno do Nordeste da Itália (Friuli) (BARBIERI, et al., 2001) e Sudeste da Califórnia (Vale Indian Wells-Owens) (GüLER, 2002) e identificar processos químicos e físicos que afetam a química da água (GüLER, THYNE, 2004).

- **Análise de imagens** - Processar imagens de ressonância magnética (AHMED, et al., 2002), comparando com diversas outras técnicas de reconhecimento de padrões (BEZDEK, et al., 1993)

- **Análise do solo** - Estudar os tipos de solo (ODEH, et al., 1992) criando inclusive uma versão aprimorada do FCM (MCBRATNEY, DEGRUIJTER, 1992).

- etc.

Existem vários trabalhos na literatura que visam aprimorar o FCM, tornando-o mais eficiente e adaptado ao problema que se deseja abordar. Estes são apenas alguns dos diversos algoritmos criados, baseados no FCM:

- literal fuzzy c-means (LFCM (BEZDEK, 1981)), que procura definir os valores iniciais do algoritmo de forma mais eficiente.

- approximate fuzzy c-means (AFCM (CANNON, et al., 1986)), que muda o algoritmo original de forma a implementá-lo de forma mais eficiente.

- fuzzy k-means with extragrades (FKME (MCBRATNEY, DEGRUIJTER, 1992)), aplicável a análise de solo.

- fast fuzzy c-means (FFCM (SHANKAR, PAL., 1994)), que faz uma abordagem voltada a grandes bancos de dados.

- conditional fuzzy c-means (CFCM (PEDRYCZ, 1996)), onde vetores de dados são agrupados sob condições baseadas em termos lingüísticos representados por conjuntos fuzzy.

- accelerated fuzzy c-means (AFCM (HERSHFINKEL, DINSTEIN, 1996)), que possui um estágio extra de aprimoramento do algoritmo FCM a cada iteração.

- fuzzy symbolic c-means (FSCM (EL-SONBATY, ISMAIL, 1998)), voltado a dados simbólicos.

- fuzzy c-medians (FCMED (KERSTEN, 1999)), que é o FCM com aproximações numéricas e normas estatísicas.

- hierarchical unsupervised fuzzy clustering (HUFC (GEVA, 1999)), que dá ao FCM o mesmo comportamento dos algoritmos de agrupamento hierárquicos, definidos no início deste capítulo.

- fuzzy J-means (FJM (BELACEL, et al., 2002)), que aborda o FCM utilizando uma nova heurística.

- alternative fuzzy c-means (AFCM (WU, YANG, 2002)), que propõe uma nova métrica de distância, em oposição à métrica Euclideana.

- bias-correct fuzzy c-means (BCFCM (AHMED, et al., 2002)), que modifica a função objetivo do FCM, para aplicação em imagens de ressonância magnética.

- extensible fast fuzzy c-means (eFFCM (PAL, BEZDEK, 2002)), voltado à problemas de tratamento de imagens, aplicando amostragem ao LFCM (BEZDEK, 1981).

- adaptive rough fuzzy leader clustering (ARFLC (ASHARAF, MURTY, 2003)), algoritmo de uma única passada, voltado para grandes bancos de dados.

- suppressed fuzzy c-means (S-FCM (FAN, et al., 2003)), algoritmo que tenta minimizar os aspectos negativos do FCM original estabelecendo uma relação natural com o K-Means.

- generalized extensible fast fuzzy c-means (geFFCM (BEZDEK, HATHAWAY, 2004)), que generaliza o algoritmo eFFCM (PAL, BEZDEK, 2002), passando a ser aplicável não só à imagens, mas a qualquer tipo de aplicação.

- fuzzy *c*-means with feature partitions (FCMP (ALEXIUK, PIZZI, 2005)), uma variante do FCM desenvolvida para integrar relacionamentos entre atributos e considerar subconjuntos de atributos do problema.

- modified suppressed fuzzy c-means (MS-FCM (HUNG, et al., 2006)), uma variante do algoritmo S-FCM, aplicável à imagens de ressonância magnética.

- etc.

Não obstante todas essas variantes, foi decidido por usar a versão básica do algoritmo FCM nesta dissertação pois, sendo a fonte inicial de onde partiram todos os outros algoritmos mencionados, podemos considerá-lo o algoritmo mais genérico, ou seja, não é voltado para nenhuma aplicação específica, podendo ser, assim, utilizado em qualquer problema. Além disso, a adaptação do método em (DOMINGOS, HULTEN, 2001) utilizou a versão original do K-Means, portanto a adaptação à versão original do algoritmo fuzzy, o FCM neste caso, seria a mais lógica e apropriada para a comparação dos resultados. E deve-se levar em conta também que, sendo a origem de todas essas outras variantes, é de se esperar que qualquer outro algoritmo ao qual o método venha a ser aplicado tenha um desempenho superior ao algoritmo criado através da adaptação do FCM. Portanto os resultados obtidos nesta dissertação tendem a ser um limite mínimo para o método. Dessa forma os experimentos descritos no Capítulo 6 se propõem a ser um guia de comparação, caso o método seja aplicado em outras variantes do FCM. Isso não seria verdadeiro em qualquer outro caso.

O FCM é tão utilizado em comparação com o K-Means pois, ao contrário deste último, o FCM é um algoritmo soft, ou seja, ele associa cada exemplo a cada cluster com um determinado grau de pertinência, e não apenas com o

cluster mais próximo. Assim, FCM define uma matriz de pertinência U, onde cada termo u_{nk} representa o grau de pertinência do exemplo n ao cluster k.

O FCM usa o parâmetro de nebulosidade m, um número real tal que $1.0 < m < \infty$. Quanto maior for m, maior a nebulosidade do agrupamento. Por outro lado, quanto mais perto m está de 1.0, o FCM fornece um resultado mais parecido com a agrupamento binário do algoritmo K-Means (BEZDEK, 1981).

Na Figura 2.6 temos o mesmo exemplo exibido anteriormente, porém ao contrário da Figura 2.5, onde m foi definido com valor $1,25$, agora utilizamos $m = 2$, o que suavizou o gráfico dos graus de pertinência, mostrado na Figura 2.7 (em cada gráfico desta figura, a linha tracejada corresponde ao cluster C_1 e a linha contínua corresponde ao cluster C_2). Nesta figura, o eixo X representa apenas os pontos pertencentes à linha central do banco de dados, ou seja, o valor 0 da Figura 2.7 representa o ponto $(0, 2)$ da Figura 2.4, o valor 1 representa o ponto $(1, 2)$, 2 representa o exemplo $(2, 2)$, e assim por diante. O eixo Y indica o valor de seus respectivos graus de pertinência.

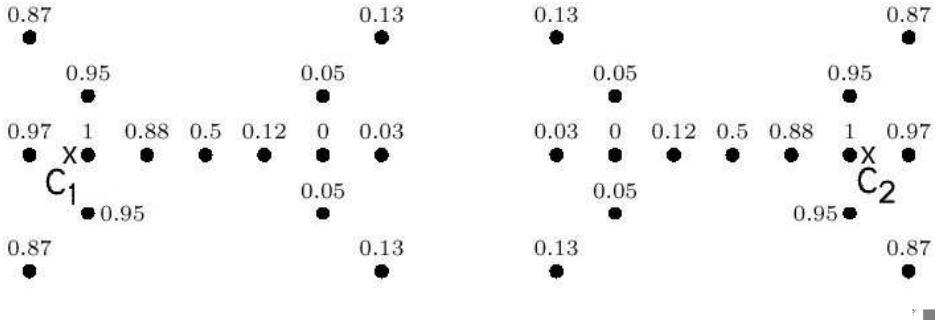

Figura 2.6: Agrupamento Fuzzy ($m = 2$) - À Esquerda os Graus de Pertinência relativos ao C_1 e à Direita os Graus de Pertinência relativos ao C_2

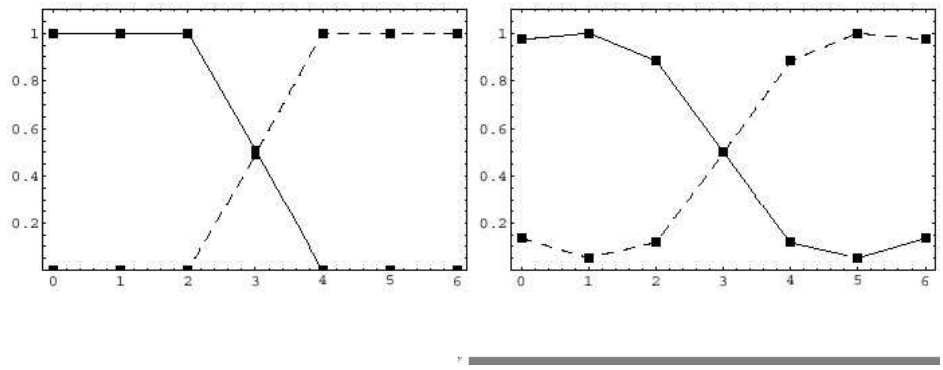

Figura 2.7: Pontos da Linha Central × Grau de Pertinência - À Esquerda Agrupamento Fuzzy com $m = 1,25$ e à Direita Agrupamento Fuzzy com $m = 2$

O algoritmo FCM, juntamente com o algoritmo K-Means, está sendo mostradon na Figura 2.8. Neste pseudo-código, na linha 4, onde é feita a atualização da matriz U, pode ser observado que usando u_{nk} como uma média fuzzy (ZADEH, 1965) ou uma escolha binária, pode-se optar entre o FCM ou o K-Means, respectivamente. Assim, este é claramente mostrado que a diferença entre o K-Means e o FCM é apenas uma fórmula; Seja $\|a - b\|$ a distância Euclideana de a até b.

Note que os valores da matriz U dependem somente de m e H (a matriz de distância entre os exemplos e os clusters). Além disso, a atualização dos clusters dependem apenas dos valores da matriz U. Essas características serão importantes futuramente.

2.3 Índices de Validação de Qualidade

Para verificar a qualidade do agrupamento, algumas medidas foram desenvolvidas (THEODORIDIS, KOUTROUMBAS, 1999; HALKIDI, et al., 2002a; HALKIDI, et al., 2002b), entre elas temos os 3 seguintes índices de

Algoritmo K/FC-Means

Entrada: K - Número de clusters

Entrada: T - Conjunto de exemplos $\{x_1, ..., x_N\}$

Entrada: m - parâmetro de nebulosidade

Entrada: γ - Limiar de convergência

Saída: C - Conjunto de centroids $\{c_1, ..., c_K\}$

Saída: U - Matriz de pertinência

1: Inicializar os centroids c_k randomicamente, $1 \leq k \leq K$

2: **repita**

3: Atualizar a matriz H onde

$$h_{nk} = \|x_n - c_k\|^2 \ \forall n, k, \ 1 \leq n \leq N, 1 \leq k \leq K$$

4: Atualizar a matriz U onde

FCM	ou	K-Means
$u_{nk} = \left[\sum_{k'=1}^{C} \left(\frac{h_{nk}}{h_{nk'}}\right)^{\frac{2}{m-1}}\right]^{-1}$		$u_{nk} = \begin{cases} 1 & \text{se } h_{nk} > h_{nk'}, k' \neq k \\ 0 & \text{caso contrário} \end{cases}$

$$\forall n, k, \quad 1 \leq n \leq N, 1 \leq k \leq K$$

5: **para** $k = 1$ to K **faça**

6: $\underline{c}_k = c_k$

7: $c_k = \frac{\sum_{n=1}^{N} u_{nk}^m x_n}{\sum_{n=1}^{N} u_{nk}^m}$

8: **fim para**

9: **até** $\sum_{k=1}^{C} (\|c_k - \underline{c}_k\|^2) < \gamma$

Figura 2.8: Algoritmo K/FC-Means

validação de qualidade de agrupamento, brevemente mostrados nessa seção: Coeficiente de Partição, Compactação/Separação e PBM(F). Eles serão utilizados no capítulo 6, onde mostramos os resultados experimentais, para comprovar a boa qualidade do algoritmo VFFCM.

2.3.1 Índice de Coeficiente de Partição

O índice de Coeficiente de Partição (PC-index (BEZDEK, et al., 1984)) é definido da seguinte forma:

$$PC = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} u_{nk}^2$$

Este índice pode assumir valores entre $1/K$ e $1,0$. Quanto mais perto o PC-index estiver de $1/K$, mais "nebuloso" será o agrupamento. Além disso, um valor próximo de $1/K$ indica que o problema é de difícil agrupamento. Este índice obtem valores máximos para partições crisp.

2.3.2 Índice de Compactação/Separação

O índice de Compactação/Separação (também conhecido como Xie-Beni index, ou somente XB (XIE, BENI, 1991)) é um índice de agrupamento fuzzy como segue:

$$XB = \frac{\text{Compactação}}{\text{Separação}}$$

onde

$$\text{Compactação} = \frac{1}{N}\sum_{n=1}^{N} u_{nk}^{m} \left\| x_n - c_k \right\|^2$$

e

$$\text{Separação} = \min_{\substack{\forall k,k' \\ 1\leq k\leq K \\ 1\leq k'\leq K \\ k\neq k'}} \left\| c_k - c_{k'} \right\|$$

Este índice torna-se mínimo para clusters compactos e bem separados. Se $m = 1,0$, então este índice pode ser usado para algoritmos crisp.

2.3.3 Índice PBM(F)

O índice PBM(F) (PAKHIRA, et al., 2004) é composto por três fatores e está definido da seguinte forma:

$$PBM(F) = \left(\frac{1}{K} \frac{E_1}{E_K} D_K \right)^2$$

onde

$$E_K = \sum_{k=1}^{K} E_k$$

$$E_k = \sum_{n=1}^{N} u_{nk}^{m} \left\| x_n - c_k \right\|$$

e

$$D_K = \max_{\substack{\forall k,k' \\ 1\leq k\leq K \\ 1\leq k'\leq K \\ k\neq k'}} \left\| c_k - c_{k'} \right\|$$

O índice PBM(F) torna-se máximo para clusters compactos e bem separados. De fato, a sigla PBM(F) está sendo usada para a especificação de

dois índices: o PBM, usado para algoritmos crisp se $m = 1,0$, e o PBMF, utilizado por algoritmos fuzzy se $m > 1,0$.

Em (PAKHIRA, et al., 2004), a superioridade do índice PBM(F) foi testada no contexto de determinar apropriadamente o número de clusters, sendo comparado aos índices Davies-Bouldin (DAVIES, BOULDIN, 1979), Dunn's (DUNN, 1973) (ambos para agrupamentos crisp) e Coeficiente de Partição (XIE, BENI, 1991), utilizando diversos conjuntos de dados reais e artificiais. PBM(F) foi melhor que os outros índices em todos os conjuntos de dados.

2.4 Amostragem Progressiva

Amostragem Progressiva (*Progressive Sampling*) é uma técnica de amostragem relativamente recente, desenvolvida em (PROVOST, et al., 1999). Ela consiste no aumento do número de exemplos, ou seja, no aumento do tamanho da partição a cada iteração, consequentemente aumentando a acurácia do modelo produzido, até que ele satisfaça o critério de convergência pretendido.

A curva de aprendizado é um gráfico que mostra a performance do modelo de acordo com o tamanho da amostra utilizado. Normalmente a curva de aprendizado possui as seguintes características:

- Uma subida íngreme em seu início.
- Uma inclinação mais suave no meio.
- Uma planície quase constante em seu final.

Um exemplo típico de uma curva de aprendizado pode ser visto na Figura 2.9, onde podemos perceber claramente os três níveis. O objetivo da

amostragem progressiva é encontrar n_{min}, ou seja, o número mínimo de exemplos necessários, quando a curva de aprendizado chega à planície. A partir de n_{min} exemplos a amostra representaria, de forma satisfatória, toda a base de dados e o aprendizado utilizando mais exemplos do banco seria desnecessário.

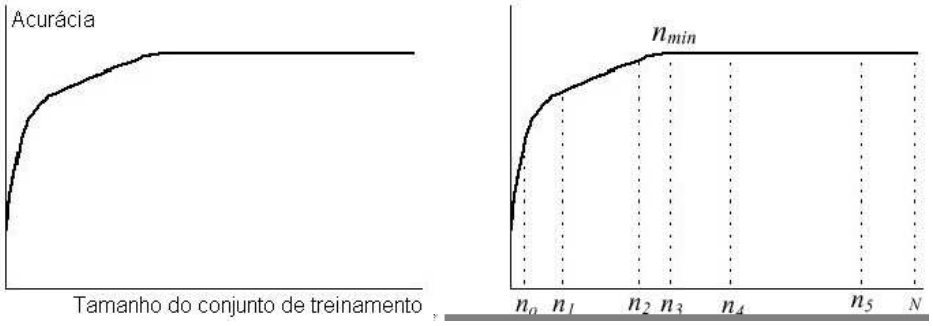

Figura 2.9: Curva de aprendizado

Determinar n_{min} é complicado pois, dependendo da aplicação, a segunda inclinação pode ser extremamente longa em algumas curvas (CATLETT, 1991a; CATLETT, 1991b) e nem mesmo existir em outras. Na Figura 2.10, os passos básicos da amostragem progressiva são apresentados em pseudo-código. Para computar os valores do conjunto S, ou seja, o número de exemplos utilizados em cada partição, utilizaremos uma progressão geométrica. Assim $S = \{n_0, an_0, a^2n_0, ..., N\}$, com constantes n_0 (valor inicial da progressão) e a (razão). Por exemplo, $S = \{100, 200, 400, 800, ..., N\}$ é uma amostragem geométrica com os valores $n_0 = 100$ e $a = 2$.

O método mais comum de seleção de quais instâncias serão usadas a cada passagem do método é a escolha aleatória. Isto é, no exemplo anterior, dos N exemplos do banco, seriam escolhidos 100 aleatoriamente para a primeira iteração do método.

Algoritmo Amostragem Progressiva

Entrada: A - Algoritmo

Entrada: T - Conjunto de exemplos $\{x_1, ..., x_N\}$

Saída: M - Modelo produzido por A utilizando n_{min} instâncias de T

1: $i = 0$

2: Computar os tamanhos das partições $S = \{n_0, n_1, n_2, ..., n_k\}$, $n_k = N$

3: **repita**

4: $n = n_i$

5: M = Modelo produzido por A utilizando n instâncias de T

6: $i++$

7: Avaliar acurácia de M

8: **até** convergir

Figura 2.10: Algoritmo Amostragem Progressiva (*Progressive Sampling*)

Capítulo 3

Very Fast K-Means

O VFKM (DOMINGOS, HULTEN, 2001), apresentado na Figura 5.1 no mesmo pseudo-código do VFFCM, é um algoritmo de agrupamento crisp que consiste de uma seqüência de rodadas do algoritmo K-Means, a cada rodada utilizando mais exemplos que a última, até que o limite de erro seja satisfeito. Este limite (obtido teoricamente pelo limite de Hoeffding (HOEFFDING, 1963)) garante que o modelo produzido pelo VFKM não difere significantemente daquele que seria obtido através da utilização de todo o banco de dados no algoritmo K-Means.

Neste capítulo, o algoritmo VFKM será apresentado, bem como suas principais características. O entendimento de seu funcionamento, e a definição de suas variáveis, que também serão descritas neste capítulo, serão importantes nos capítulos seguintes desta dissertação.

Os seguintes passos serão seguidos para demonstrar o funcionamento do algoritmo VFKM: primeiro, serão mostradas as duas fontes de erro existentes, causadas pela utilização de um menor número de exemplos (amostragem), ao invés da base de dados inteira. A seguir, as variáveis do algoritmo que representam essas fontes de erro serão definidas. Depois disso, o limite de

Hoeffding será revisado. As variáveis do algoritmo que foram definidas serão, então, associadas ao limite de Hoeffding. O próximo passo será mostrar a função de perda, que também estará associada ao limite Hoeffding. Esta função será usada para calcular o limite de erro do algoritmo. Finalmente, o último passo, será mostrar as equações do VFKM (erro e número de exemplos), que foram obtidas em (DOMINGOS, HULTEN, 2001).

3.1 Fontes de Erro a Cada Iteração

Utilizar amostragem em um algoritmo de aprendizado de máquinas cria duas fontes de erro a cada iteração:

- Erro de amostragem → Fonte de erro causada pelo número *finito* de exemplos utilizados em uma determinada iteração.
- Erro de atribuição → Fonte de erro devido ao fato das *posições iniciais dos centroids em cada iteração não serem as corretas* (ou seja, as que seriam obtidas com infinitos exemplos).

Note que o erro de amostragem existe somente porque utilizamos menos exemplos *em uma determinada iteração*. Contudo, o erro de atribuição de uma iteração é em função do erro da iteração anterior, já que os centroids, no começo de alguma iteração, têm suas posições definidas de acordo com o cálculo da iteração anterior.

3.2 Definindo as Variáveis no Algoritmo

Considerando que c é usado como uma variável que define um cluster do algoritmo, nós usamos a Tabela 3.1 para seus índices.

Variável	Domínio	Definição
k	de 1 a K	número de clusters
d	de 1 a D	número de dimensões do problema
i	de 1 a I	iteração

Tabela 3.1: Índices do Cluster

Assim, c_{kdi} é a d-ésima dimensão (ou coordenada, ou atributo) do k-ésimo(a) centroid na iteração i. De forma similar, c_{ki} é o vetor do k-ésimo(a) centroid na iteração i. Agora nós podemos inferir as definições da Tabela 3.2.

Variável	**d-ésima coordenada do(a) k-ésimo centroid na i-ésima iteração...**
c_{kdi}	usando *infinitos* exemplos
$\hat{c}_{kdi}$	usando *finitos* exemplos
$\bar{c}_{kdi}$	usando *finitos* exemplos e *sem erro de atribuição*

Tabela 3.2: Nomenclatura das Variáveis de Cluster

Usando a Tabela 3.2 nós podemos definir a Tabela 3.3, onde nós explicamos as combinações entre os diferentes tipos de clusters (todos os sub-índices foram omitidos por praticidade).

Variável	Descrição
c	c (centroid sem erro)
$\hat{c}$	c + erro de amostragem + erro de atribuição
$\bar{c}$	c + erro de amostragem
$\lvert\bar{c} - c\rvert$	erro de amostragem
$\lvert\hat{c} - \bar{c}\rvert$	erro de atribuição
$\lvert\hat{c} - c\rvert$	erro da iteração = erro de amostragem + erro de atribuição

Tabela 3.3: Descrição das Variáveis de Cluster em Relação aos Erros

3.3 Limite de Hoeffding

Seja r uma variável randômica com domínio de tamanho R. Suponha que devemos fazer n observações independentes de r e calcular sua média $\bar{r}$. O *limite de Hoeffding* (HOEFFDING, 1963) (também conhecido com limite de Chernoff aditivo) nos diz que: $P(\bar{r} - \epsilon \leq \mu_r \leq \bar{r} + \epsilon) = 1 - \delta$, onde

- μ_r é a verdadeira média de r
- $\epsilon = \sqrt{\frac{R^2 \ln\left(\frac{2}{\delta}\right)}{2n}}$
- δ é um número muito baixo

Assim, de acordo com o limite de Hoeffding, μ_r tem grande probabilidade de estar dentro do intervalo da Figura 3.1, ou seja, a média $\bar{r}$ calculada não é a verdadeira média μ_r, mas com uma probabilidade $(1-\delta)$ (que nós definimos alta), $\bar{r}$ possui um erro ϵ conhecido e esse erro é inversamente proporcional ao número de observações n. Assim, enquanto n aumenta, o erro ϵ diminui (considerando δ constante).

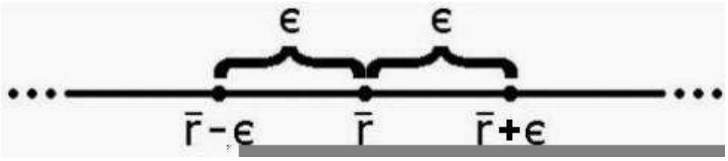

Figura 3.1: Intervalo de erro do limite de Hoeffding

O limite de Hoeffding é um resultado estatístico independente da distribuição de probabilidade gerada pelas n observações feitas. Essa característica tem a desvantagem de necessitar de mais observações para chegar a mesma probabilidade δ e o mesmo erro ϵ. Porém sua grande vantagem é a genera-

lidade que esse limite fornece, permitido que ele seja aplicado em diversas situações sem se preocupar com a distribuição existente.

3.4 Associando as Variáveis de Hoeffding com as Variáveis do Algoritmo

Para que todo processo possa ser entendido, iremos associar as variáveis de Hoeffding com as variáveis do algoritmo previamente definidas na seção 3.2. Cada cluster c_{kdi} é uma variável randômica r especificada na seção 3.3. O erro ϵ de cada uma das variáveis c_{kdi} é definida como ϵ_{kdi}, que será a soma dos erros de amostragem e de atribuição. De acordo com o limite de Hoeffding, $\hat{c}_{kdi} - \epsilon_{kdi} \leq c_{kdi} \leq \hat{c}_{kdi} + \epsilon_{kdi}$, ou seja, o verdadeiro centroid c_{kdi} (que seria gerado utilizando-se infinitos exemplos) está dentro de um intervalo limitado pelo erro ϵ_{kdi} (ainda desconhecido) com probabilidade $(1 - \delta)$, definindo δ com um valor muito baixo.

Note que cada associação feita está relacionada a cada dimensão d do cluster k na iteração i, então é necessário que exista um outro limite, que será o *limite de Hoeffding total*, para englobar todas essas variáveis. Este limite de Hoeffding total possui um *erro total* ϵ^* e uma *probabilidade total* δ^*, onde $(1 - \delta^*) = (1 - \delta)^{KDI} \Rightarrow \delta = 1 - \sqrt[KDI]{1 - \delta^*}$, (na pior hipótese) assumindo independência. O erro total ϵ^* será o limite máximo para a função de perda $L(C_{\vec{n}}, C_\infty)$, que será definida na próxima seção.

3.5 Função de Perda

Comparando dois resultados finais C_1 e C_2, fornecidos por um determinado algoritmo, ambos com o mesmo número de clusters K, definimos a *função de perda* como a soma das distâncias entre seus clusters:

$$L(C_1, C_2) = \sum_{k=1}^{K} \|c_{1,k} - c_{2,k}\|^2. \tag{3.1}$$

Nós a usaremos para comparar dois diferentes agrupamentos fornecidos pelo algoritmo FCM, utilizando o mesmo número de clusters K e o mesmo parâmetro de nebulosidade m.

Seja n_i o número de exemplos usados em cada iteração i do algoritmo. O objetivo é reduzir n_i até que a função de perda $L(C_{\vec{n}}, C_\infty)$ alcance o erro total ϵ^* citado na seção anterior, onde $C_{\vec{n}}$ é a solução utilizando finitos exemplos, C_∞ é a solução com infinitos exemplos. Assim, ϵ^* será o limite máximo da função de perda, sendo essa função o somatório de todos os erros na última iteração I, ou seja:

$$L(C_{\vec{n}}, C_\infty) = \sum_{k=1}^{K} \sum_{d=1}^{D} \epsilon_{kdI}^2 \leq \epsilon^* \tag{3.2}$$

3.6 Equações de Erro e Número de Exemplos

As fórmulas utilizadas para os cálculos de ϵ_{kdi} e do número de exemplos n_i, utilizado a cada iteração i do algoritmo K-Means, são as que seguem e foram desenvolvidas matematicamente em (DOMINGOS, HULTEN, 2001).

$$\epsilon_{kdi} = \left| \frac{\max\left(\sum_{x|\Delta x_{kdi}>0} \Delta x_{kdi}, \sum_{x|\Delta x_{kdi}<0} |\Delta x_{kdi}|\right)}{\hat{n}_{ki} - \tilde{n}^{+}_{ki}} \right| + \sqrt{\frac{R_d^2 \ln\left(\frac{2}{\delta}\right)}{2\left(\hat{n}_{ki} - \tilde{n}^{+}_{ki}\right)}} \tag{3.3}$$

onde

$\hat{n}_{ki}$ =número de exemplos que $\hat{c}_{ki}$ "ganhou"

$\hat{n}^{+}_{ki}$ =número de exemplos de S^{+}_{ki}

$\hat{n}^{-}_{ki}$ =número de exemplos de S^{-}_{ki}

$\tilde{n}^{+}_{ki}$ =limite superior do $\hat{n}^{+}_{ki}$

$$\Delta x_{kdi} = \begin{cases} x_{nd} - \hat{c}_{kdi} & \text{se } x_n \in S^{+}_{ki} \\ -(x_{nd} - \hat{c}_{kdi}) & \text{se } x_n \in S^{-}_{ki} \end{cases}$$

x_n =n-ésimo exemplo

e

x_{nd} =d-éssima coordenada do n-ésimo exemplo

As variáveis S^{+}_{ki} e S^{-}_{ki} definem os conjuntos de exemplos que satisfazem, respectivamente, às condições (4.1) e (4.2). Mais detalhes sobre essas duas equações serão mostrados no próximo capítulo.

Por último, falta mostrar o número de exemplos usados, a cada iteração, de uma rodada do algoritmo K-Means, que será definido pela equação:

$$n_i = \max_k \left(\frac{\hat{n}_{ki}}{f_{ki}}\right) \tag{3.4}$$

onde

$$
\begin{aligned}
f_{ki} =& \frac{\hat{n}_{ki}}{n_i} \\
\hat{n}_{ki} =& \left(\frac{\sum_{j=1}^{I} \sqrt[3]{r_{ki} r_{kj}^2}}{\sqrt{\frac{\epsilon^*}{K}} + r_k} \right)^2 \\
r_{ki} =& \sqrt{\frac{R^2 \ln\left(\frac{2}{\delta}\right)}{2\left(1 - b_{ki}\epsilon_{k,i-1}\right)}} \prod_{j=i+1}^{I} \alpha_{kj} \\
r_k =& \sum_{i=1}^{I} \left[\frac{a_{ki} b_{ki} \left(\epsilon_{k,i-1}\right)^2}{\left(1 - b_{ki}\epsilon_{k,i-1}\right)^2} \prod_{j=i+1}^{I} \alpha_{kj} \right] \\
\alpha_{ki} =& \frac{a_{ki}}{\left(1 - b_{ki}\epsilon_{k,i-1}\right)^2} \\
a_{ki} =& \frac{\sqrt{\sum_{d=1}^{D} X_{kdi}^2}}{\hat{n}_{ki}\epsilon_{k,i-1}} \\
b_{ki} =& \frac{\tilde{n}_{ki}^{+}}{\hat{n}_{ki}\epsilon_{k,i-1}}
\end{aligned}
$$

and

$$
X_{kdi} = \max\left(\sum_{x|\Delta x_{kdi}>0} \Delta x_{kdi}, \sum_{x|\Delta x_{kdi}<0} |\Delta x_{kdi}| \right)
$$

Estas duas fórmulas serão utilizadas no algoritmo VFKM, que é mostrado na Figura 5.1 juntamente com o VFFCM. Na seção 5.3 este pseudo-código é descrito em mais detalhes.

Capítulo 4

Metodologia Fuzzy

Pode ser percebido sem muita dificuldade que, com exceção das fórmulas de erro e do número de exemplos, todas as outras definições feitas no capítulo anterior podem ser generalizadas para qualquer algoritmo fuzzy. Para isso, basta levar em conta que os clusters seriam, na realidade, as classes do algoritmo de aprendizado de máquinas soft ao qual o método seria aplicado. Assim, todas as definições feitas (como as fontes de erro, as associações com o limite de Hoeffding e a função de perda) podem ser utilizadas para a adaptação do método, que havia sido aplicado ao algoritmo K-Means, para algoritmos fuzzy em geral. Em particular, a adaptação do método ao algoritmo fuzzy FCM será feita no capítulo seguinte.

Neste capítulo, o método geral para tornar um algoritmo fuzzy escalável é apresentado, além das novas definições de *fuzzy sampling false positives* e *fuzzy sampling false negatives*, cruciais para a aplicação desse método à esse tipo de algoritmo.

4.1 O Método

Seja A um algoritmo iterativo de aprendizado de máquina fuzzy com $st_1, st_2, \ldots, st_i, \ldots$ passos. Assumindo um conjunto de treinamento T com N exemplos i.i.d. (independentes e identicamente distribuídos) como entrada de A. Seja $n_i \leq N$ o número de exemplos usados em cada passo, $\vec{n} = \{n_1, n_2, \ldots, n_i, \ldots\}$ e seja δ_i a probabilidade de fazer a escolha errada no passo st_i usando n_i exemplos, produzindo um erro ϵ_i. Utilizando o limite de Hoeffding (ou outro limite estatístico similar) podemos expressar ϵ_i e δ_i em função de n_i.

Seja M_∞ o modelo produzido por A usando infinitos exemplos a cada passo, isto é, $\forall i \quad n_i = \infty$, e seja $M_{\vec{n}}$ o modelo produzido usando n_i exemplos no passo st_i. Seja $L(M_1, M_2)$ a diferença entre os modelos M_1 e M_2, cada um deles utilizando dois diferentes vetores $\vec{n_1}$ e $\vec{n_2}$.

Precisa-se obter um limite para $L(M_{\vec{n}}, M_\infty)$ em função de δ_i e ϵ_i, e assim em função de n_i, para todos os passos de A: $L(M_{\vec{n}}, M_\infty) \leq G_{A,T}(\vec{n})$ com probabilidade de pelo menos $1 - F_{A,T}(\vec{n})$, onde F e G são funções que dependem do algoritmo fuzzy A e do conjunto de treinamento T.

Levando em consideração o limite de Hoeffding, isto é, definindo que $L(M_{\vec{n}}, M_\infty) \leq \epsilon$ com probabilidade $(1 - \delta)$, podemos ver que se $G_{A,T}(\vec{n}) \leq \epsilon$ e $F_{A,T}(\vec{n}) \leq \delta$ então $\vec{n}$ é suficientemente grande, caso contrário A precisa utilizar mais exemplos. Inversamente, podemos determinar $\vec{n}$ utilizando $G_{A,T}(\vec{n})$ e $F_{A,T}(\vec{n})$ para reduzir o tempo de execução de A; é desta maneira que o método é aplicado, calculando a função $G_{A,T}(\vec{n})$, que será o erro do algoritmo, e determinando assim $\vec{n}$. A função $F_{A,T}(\vec{n})$ já é previamente definida no início do algoritmo como uma probabilidade máxima desejada.

Porém nesse novo método que estamos desenvolvendo, a função $G_{A,T}(\vec{n})$ e, consequentemente, o valor de $\vec{n}$, são calculados, ao contrário de (DO-

MINGOS, HULTEN, 2001), baseados nas nossas novas definições de *fuzzy sampling false positives* e *fuzzy sampling false negatives*. Esses dois novos conceitos são calculados utilizando a matriz de pertinência, uma estrutura inerente aos algoritmos de apredizado de máquina fuzzy. As novas definições, que serão explicadas na próxima seção, são essenciais para aplicar a metodologia com sucesso nos algoritmos fuzzy, porque permitem fazer os cálculos necessários do método de maneira soft.

4.2 *Fuzzy Sampling False Positives* e *Fuzzy Sampling False Negatives*

Para entender como o método pode ser aplicado ao FCM e genericamente em outros algoritmos fuzzy, nós primeiramente definimos, no VFKM (DOMINGOS, HULTEN, 2001), os exemplos *sampling false positives* e *sampling false negatives*, com respeito a um cluster, que são causados pelo uso de menos exemplos que os dados infinitos disponíveis. Suponha que em uma iteração i, x_n (n-ésimo exemplo do banco de dados) é "ganho" pelo centroid $\hat{c}_{ki}$, que está a uma distância d_{ki} do mesmo. Se existir um outro centroid $\hat{c}_{k'i}$, a uma distância $d_{k'i}$ dele tal que

$$d_{k'i} - \epsilon_{k'i} < d_{ki} + \epsilon_{ki}, \tag{4.1}$$

então o exemplo x_n talvez tenha sido incorretamente associado a $\hat{c}_{ki}$. Nós chamamos o exemplo x_n um sampling false positive com respeito ao centroid $\hat{c}_{ki}$. Podemos verificar um exemplo sampling false positive na Figura 4.1. Na linha de cima o erro não foi considerado, porém na linha de baixo, considerando os erros introduzidos pelo limite de Hoeffding (mais afastado do cluster que o "ganhou" e mais próximo dos outros clusters), o exemplo x_n passa a

ser associado à um outro centroid. Ou seja, ele foi "ganho" por um cluster que não deveria ter sido, por isso ele é um sampling false positive relativo à este cluster.

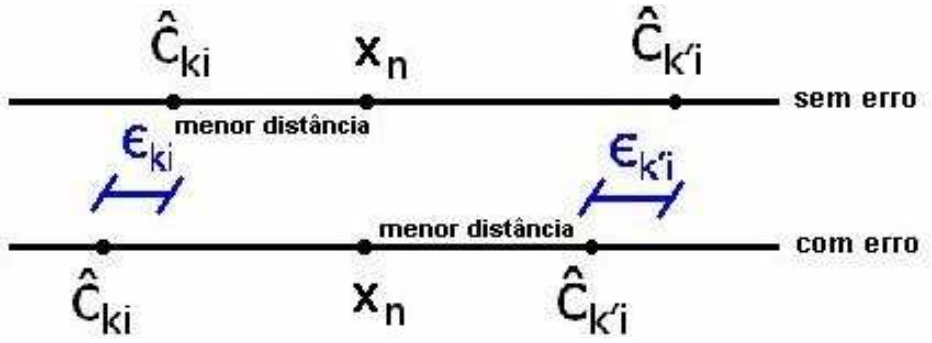

Figura 4.1: Exemplo x_n sampling false positive

De forma similar, nós chamamos o exemplo x_n um sampling false negative com respeito ao centroid $\hat{c}_{ki}$ se x_n, na iteração i, foi "ganho" por $\hat{c}_{k'i}$ à uma distância $d_{k'i}$ dele, mas deveria ter sido "ganho" por $\hat{c}_{ki}$, distante d_{ki}, tal que

$$d_{ki} - \epsilon_{ki} < d_{k'i} + \epsilon_{k'i}. \tag{4.2}$$

Novamente podemos verificar, através da Figura 4.2, um exemplo sampling false negative. Considerando o erro do limite de Hoeffding (linha de baixo), ou seja, levando em consideração que o exemplo deveria estar mais afastado do cluster que o "ganhou" e mais próximo aos outros clusters, o exemplo terminou por ser associado à outro cluster, que não havia sido associado sem levar o erro em consideração. Portanto este exemplo é um sampling false negative relativo ao cluster que ao qual ele deveria ter sido associado ($\hat{c}_{ki}$).

Assim, nos dois casos, para entender porque o exemplo x_n provavelmente tenha sido erroneamente "ganho" por algum cluster, em (4.1) e em (4.2)

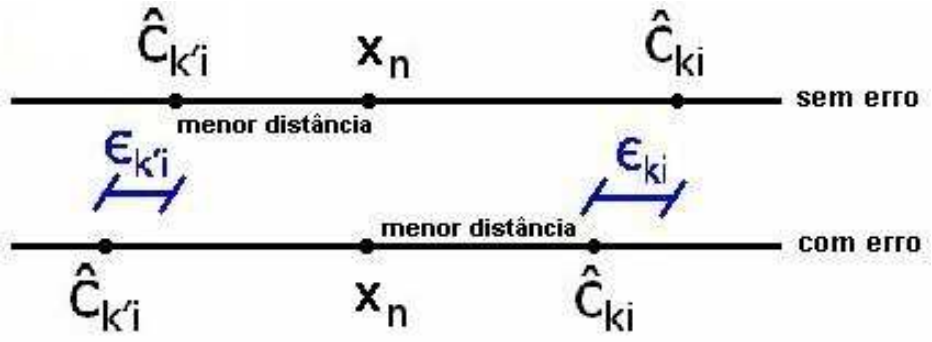

Figura 4.2: Exemplo x_n sampling false negative

o centroid mais próximo passou a estar o mais longe possível, e os outros centroids passaram a estar o mais perto possível de x_n, de acordo com o erro do limite de Hoeffding.

Em (DOMINGOS, HULTEN, 2001), o conjunto de todos os sampling false positives (negatives) é denotado por S^+_{ki} (S^-_{ki}) e a soma do número de exemplos desse conjuntos é dada por n^+_{ki} (n^-_{ki}). Esses valores, já citados nas fórmulas da seção 3.6, foram importantes para o desenvolvimento do método desenvolvido naquele trabalho.

Entretanto, se levarmos em conta um algoritmo fuzzy, e, consequentemente, não apenas algoritmos de agrupamento, mas também problemas de classificação fuzzy, um(a) cluster/classe não "ganha" um exemplo x_n, porque todos(as) clusters/classes possuem uma determinada associação com cada um deles(as) de acordo com a matriz de pertinência U. Então, não podemos chamar x_n de um exemplo sampling false positive. Ao invés disso, chamamos cada x_n de um exemplo *fuzzy sampling false positive* (*fsfp*) com respeito a centroid/classe $\hat{c}_{ki}$ e *com grau* $s_{nk,i}$, se $s_{nk,i}$ é um termo positivos de S, onde $s_{nk,i} = u_{nk} - u'_{nk}$ (subtração de U e U'). U' é calculado considerando os possíveis erros de cada centroid que são calculados utilizando a nova matriz

temporária H', que é baseada na matriz H do algoritmo FCM (matriz de distância entre os exemplos e os clusters), mas considerando o erro do limite de Hoeffding:

$$h'_{nk} = \begin{cases} \|x_n - \hat{c}_{ki}\| + \epsilon_{ki} & \text{se } \hat{c}_{ki} \text{ é o cluster mais próximo de} \\ & x_n \text{ na iteração } i \\ \|x_n - \hat{c}_{ki}\| - \epsilon_{ki} & \text{caso contrário} \end{cases}$$

A matriz U do FCM depende unicamente do valor de m e da matriz de distância H; como nossa segunda matriz auxiliar U', mas utilizando nossa nova matriz H' ao invés de H em sua fórmula: $u'_{nk} = \left[\sum_{j=1}^{C} \left(\frac{h'_{nk}}{h'_{nj}}\right)^{\frac{2}{m-1}}\right]^{-1}$.

De forma análoga ao *fsfp*, nós chamamos cada exemplo x_n um *fuzzy sampling false negative* (*fsfn*) com respeito a centroid/classe $\hat{c}_{ki}$ e *com grau* $s_{nk,i}$, se $s_{nk,i}$ é um termo negativo de S.

Essas definições serão utilizadas para aplicar o método ao algoritmo FCM no próximo capítulo, mas, genericamente, podem ser adaptadas para qualquer algoritmo de aprendizado de máquina fuzzy que utiliza uma matriz de pertinência.

Assim, depois de explicitadas todas essas definições, podemos colocar o método, em linhas gerais, da forma descrita na Figura 4.3, em pseudo-código. É importante notar que, apesar de A ser o algoritmo original ao qual o método está sendo aplicado, provavelmente algumas modificações no mesmo serão necessárias para que ele possa fornecer os dados que permitam calcular o erro de amostragem, os *fsfp* e os *fsfn*. Esses cálculos serão necessários para definir o vetor de exemplos $\vec{n}$ para a próxima rodada do método.

No capítulo seguinte serão mostrados os cálculos do erro de amostragem e do vetor $\vec{n}$, considerando como A, o algoritmo FCM.

Algoritmo Método Geral

Entrada: A - Algoritmo

Entrada: T - Conjunto de exemplos $\{x_1, ..., x_N\}$

Saída: M - Modelo produzido por A utilizando n_i instâncias de T a cada iteração i de A

1: **repita**

2: Calcular $\vec{n} = \{n_1, n_2, ...\}$

3: Executar A utilizando n_i exemplos a cada iteração i

4: Calcular o erro de amostragem usando o limite de Hoeffding e os *fsfp* e *fsfn*

5: **até** alcançar o limite de erro

Figura 4.3: Método Geral

Capítulo 5

Very Fast Fuzzy C-Means

Nesta seção, o método desenvolvido no capítulo anterior será aplicado no algoritmo FCM, resultando em um novo algoritmo denominado Very Fast Fuzzy C-Means (VFFCM). Para isso, precisamos calcular o erro da iteração ϵ_{kdi}, que nos fornece o limite superior da perda $L(C_{\vec{n}}, C_\infty)$ e o número de exemplos n_i de cada iteração.

5.1 Calculando o Erro

Para calcular o erro de iteração ϵ_{kdi} da Tabela 3.3, que é a soma dos erros de amostragem e de atribuição: $\epsilon_{kdi} = |\hat{c}_{kdi} - c_{kdi}| = |\hat{c}_{kdi} - \bar{c}_{kdi}| + |\bar{c}_{kdi} - c_{kdi}|$ precisamos colocá-lo em função de $\hat{c}_{kdi}$, *fsfp* e *fsfn*.

O número de exemplos n_i de uma iteração é igual a soma de todos os termos de U, i. e., $n_i = \sum_{k=1}^{K} \sum_{n=1}^{N} u_{nk}^m$ para todo i. Portanto, nós definimos o número de exemplos associado a cada cluster $\hat{c}_{ki}$ como $\sum u_{k,i} = \sum_{n=1}^{N} u_{nk}^m$. Para definir n', o número de exemplos que o cluster $\hat{c}_{ki}$ deveria ter, nós precisamos subtrair de $\sum u_{k,i}$ o total de *fsfp* e o total de *fsfn* relativo ao cluster $\hat{c}_{ki}$, que é definido por:

- $\sum u^{+}_{k,i} = \sum_{n=1}^{n_i} s^{m}_{nk,i}, \forall s_{nk,i} \geq 0$
- $\sum u^{-}_{k,i} = |\sum_{n=1}^{n_i} s^{m}_{nk,i}|, \forall s_{nk,i} < 0$

Então, para obter o limite superior do erro de amostragem, note que esse erro depende unicamente do uso de finitos exemplos na iteração i. Deste modo, o erro de amostragem vem diretamente do erro ϵ do limite de Hoeffding, usando n':

$$|\bar{c}_{kdi} - c_{kdi}| \leq \sqrt{\frac{R_d^2 \ln\left(\frac{2}{\delta}\right)}{2n'}} = \sqrt{\frac{R_d^2 \ln\left(\frac{2}{\delta}\right)}{2\left(\sum u_{k,i} - \sum u^{+}_{k,i} + \sum u^{-}_{k,i}\right)}}$$

onde R_d é o domínio da d-ésima dimensão do problema.

Seja $\sum \tilde{u}^{+}_{k,i}$ o limite superior de $\sum u^{+}_{k,i}$, então, $0 \leq \sum \tilde{u}^{+}_{k,i} \leq \sum u^{+}_{k,i}$. Assim, temos $\sum u_{k,i} - \sum u^{+}_{k,i} + \sum u^{-}_{k,i} \geq \sum u_{k,i} - \sum \tilde{u}^{+}_{k,i}$. Então, obtemos o limite superior do erro de amostragem:

$$|\bar{c}_{kdi} - c_{kdi}| \leq \sqrt{\frac{R_d^2 \ln\left(\frac{2}{\delta}\right)}{2\left(\sum u_{k,i} - \sum \tilde{u}^{+}_{k,i}\right)}} \tag{5.1}$$

O erro de atribuição é: $|\hat{c}_{kdi} - \bar{c}_{kdi}|$. Usando a Tabela 3.3, pode-se ver que o erro de atribuição é $|\hat{c}_{kdi} - (\hat{c}_{kdi}-\text{assigment error})|$. Então, temos que subtrair o erro de atribuição do $\hat{c}_{kdi}$. Já que $\hat{c}_{kdi}$ é computado usando o passo 7 do algoritmo da Figura 2.8 como $\hat{c}_k = \frac{\sum_{n=1}^{N} u^{m}_{nk} x_n}{\sum_{n=1}^{N} u^{m}_{nk}}$, precisamos subtrair o erro de atribuição no numerador e no denominador. O numerador de $\hat{c}_{kdi}$ é calculado multiplicando cada exemplo x_{nk} pelo seu respectivo termo u_{nk} da matriz U relativo ao k-ésimo cluster. Para subtrair o erro de atribuição, temos que fazer o mesmo cálculo, porém agora utilizando $s_{nk,i-1}$ ao invés de u_{nk}. Este cálculo corresponde ao numerador da fórmula de $\hat{c}_{kdi}$, mas nos fornece o erro que os *fsfp* e *fsfn* colocaram até a $(i-1)$-ésima iteração, nos dando assim o erro de atribuição da iteração i. Logo, definimos:

- $\sum \left(u^{+}_{k,i} x_{d,i}\right) = \sum_{n=1}^{n_i} (s^{m}_{nk,i} x_{nd}), \forall s_{nk,i} \geq 0$ (valor excedente no numerador da fórmula do cluster $\hat{c}_{ki}$)

- $\sum \left(u^{-}_{k,i} x_{d,i}\right) = \sum_{n=1}^{n_i} (|s^{m}_{nk,i}| x_{nd}), \forall s_{nk,i} \geq 0$ (valor faltando no numerador da fórmula do cluster $\hat{c}_{ki}$)

onde x_{nd} é a d-ésima dimensão do n-ésimo exemplo.

Fazendo o mesmo no denominador da fórmula de $\hat{c}_{kdi}$, computamos o erro de atribuição:

$$|\hat{c}_{kdi} - \bar{c}_{kdi}| =$$

$$\left| \hat{c}_{kdi} - \frac{\left(\sum u_{k,i}\right) \hat{c}_{kdi}}{\sum u_{k,i} - \sum u^{+}_{k,i} + \sum u^{-}_{k,i}} - \frac{\sum \left(u^{+}_{k,i} x_{d,i}\right)}{\sum u_{k,i} - \sum u^{+}_{k,i} + \sum u^{-}_{k,i}} + \frac{\sum \left(u^{-}_{k,i} x_{d,i}\right)}{\sum u_{k,i} - \sum u^{+}_{k,i} + \sum u^{-}_{k,i}} \right| =$$

$$= \left| \frac{X_{kdi}}{\sum u_{k,i} - \sum u^{+}_{k,i} + \sum u^{-}_{k,i}} \right|$$

onde $X_{kdi} = \left(\sum u^{-}_{k,i} - \sum u^{+}_{k,i}\right) \hat{c}_{kdi} - \sum \left(u^{-}_{k,i} x_{d,i}\right) + \sum \left(u^{+}_{k,i} x_{d,i}\right)$.

Para obtermos o limite superior para esta equação, calculamos o limite inferior de seu denominador. Utilizando $\sum \tilde{u}^{+}_{k,i}$ definido anteriormente, obtemos o limite superior do erro de atribuição:

$$|\hat{c}_{kdi} - \bar{c}_{kdi}| \leq \left| \frac{X_{kdi}}{\sum u_{k,i} - \sum \tilde{u}^{+}_{k,i}} \right| \tag{5.2}$$

Logo, o limite superior do erro da iteração ϵ_{kdi} é:

$$|\hat{c}_{kdi} - c_{kdi}| \leq \text{Equação (5.1)} + \text{Equação (5.2)} = \epsilon_{kdi} \tag{5.3}$$

Utilizando este ϵ_{kdi}, o limite superior da função de perda é obtido como na Inequação (3.2). isto garante que VFFCM produz o mesmo modelo de agrupamento que seria encontrado com infinitos dados, com uma probabilidade $(1 - \delta^{*})$ alta.

5.2 Calculando o Número de Exemplos a Cada Iteração

O número de exemplos n_i, para todo i, precisa ser minimizado, submetido à restrição $\sum_{k=1}^{K} \|\hat{c}_{kI} - c_{kI}\|^2 \leq \epsilon^*$. Já que $n_i = \sum_{k=1}^{K} \sum_{n=1}^{N} u_{nk}$ para cada i, então $\sum u_{k,i}$ necessita ser minimizado.

Uma condição suficiente para $\sum_{k=1}^{K} \|\hat{c}_{kI} - c_{kI}\|^2 \leq \epsilon^*$ é $\|\hat{c}_{kI} - c_{kI}\| \leq \sqrt{\epsilon^*/K}$ para todo k. Então $\|\hat{c}_{kI} - c_{kI}\|$ precisa estar em função de $\sum u_{k,i}$.

Da Equação (5.1) obtemos:

$$|\bar{c}_{ki} - c_{ki}| \leq$$

$$\sqrt{\sum_{d=1}^{D} \frac{R_d^2 \ln\left(\frac{2}{\delta}\right)}{2\left(\sum u_{k,i} - \sum \tilde{u}_{k,i}^{+}\right)}} = \sqrt{\frac{R^2 \ln\left(\frac{2}{\delta}\right)}{2\left(\sum u_{k,i} - \sum \tilde{u}_{k,i}^{+}\right)}}$$

onde $R^2 = \sum_{d=1}^{D} R_d^2$.

Dada a Equação (5.2) temos:

$$|\hat{c}_{ki} - \bar{c}_{ki}| \leq \left| \frac{\sqrt{\sum_{d=1}^{D} X_{kdi}^2}}{\sum u_{k,i} - \sum \tilde{u}_{k,i}^{+}} \right| .$$

Seja a_{ki} e b_{ki}:

$$a_{ki} = \frac{\sqrt{\sum_{d=1}^{D} X_{kdi}^2}}{\sum u_{k,i}\epsilon_{k,i-1}} \qquad \text{e} \qquad b_{ki} = \frac{\sum \tilde{u}_{k,i}^{+}}{\sum u_{k,i}\epsilon_{k,i-1}}$$

Substituindo-os no erro de iteração ϵ_{ki} (Equação (5.3)), temos

$$\|\hat{c}_{ki} - c_{ki}\| \leq$$

$$\frac{a_{ki}\epsilon_{k,i-1}}{1 - b_{ki}\epsilon_{k,i-1}} + \sqrt{\frac{R^2 \ln\left(\frac{2}{\delta}\right)}{2\sum u_{k,i}\left(1 - b_{ki}\epsilon_{k,i-1}\right)}} = \epsilon_{ki} \tag{5.4}$$

A Equação (5.4) é a Equação (12) em (DOMINGOS, HULTEN, 2001) com $\hat{n}_{ki}$ ao invés de $\sum u_{k,i}$, a variável desconhecida.

Aproximando ϵ_{ki} pelos dois primeiros termos dessa Expressão de Taylor, expandindo em torno do ponto $\epsilon^0_{k,i-1}$ e discartando os termos da primeira derivada que se tornam desprezíveis a medida que $\sum u_{k,i}$ aumenta, obtemos

$$\epsilon_{ki} = \alpha_{ki}\epsilon_{k,i-1} + \beta_{ki}$$

onde

$$\alpha_{ki} = \frac{a_{ki}}{1 - b_{ki}{\epsilon^0_{k,i-1}}^2}$$

e

$$\beta_{ki} = \frac{a_{ki}\epsilon^0_{k,i-1}}{1 - b_{ki}\epsilon^0_{k,i-1}} + \sqrt{\frac{R^2 \ln\left(\frac{2}{\delta}\right)}{2\sum u_{ki}\left(1 - b_{ki}\epsilon^0_{k,i-1}\right)}} - \alpha_{ki}\epsilon^0_{k,i-1}$$

Sendo ϵ_{k0}, podemos mostrar por indução que:

$$\epsilon_{km} = \sum_{i=1}^{I} \beta_{ki} \prod_{j=i+1}^{I} \alpha_{kj} = \sum_{i=1}^{I} \frac{r_{ki}}{\sqrt{\sum u_{ki}}} - r_k$$

onde

$$r_{ki} = \sqrt{\frac{R^2 \ln\left(\frac{2}{\delta}\right)}{2\left(1 - b_{ki}\epsilon^0_{k,i-1}\right)}} \prod_{j=i+1}^{I} \alpha_{kj} \qquad \text{e} \qquad r_k = \sum_{i=1}^{I} \left[\frac{a_{ki}b_{ki}\left(\epsilon^0_{k,i-1}\right)^2}{\left(1 - b_{ki}\epsilon_{k,i-1}\right)^2} \prod_{j=i+1}^{I} \alpha_{kj} \right]$$

Finalmente podemos minimizar $\sum u_{k,i}$ submetido à $\epsilon_{kI} = \sqrt{\frac{\epsilon^*}{K}}$ aplicando o método dos multiplicadores de Lagrange na função Lagrangeana abaixo:

$$L(\overrightarrow{\sum u_{ki}}, \lambda) = \sum_{i=0}^{I} \sum u_{ki} + \lambda \left(\sum_{i=1}^{I} \frac{r_{ki}}{\sqrt{\sum u_{ki}}} - r_k - \sqrt{\frac{\epsilon^*}{k}} \right)$$

Igualando o gradiente de $L(\overrightarrow{\sum u_{ki}}, \lambda)$ a zero com respeito a $\sum u_{ki}$ e λ e colocando $\sum u_{ki}$ em evidência, temos:

$$\sum u_{k,i} = \left(\frac{\sum_{j=1}^{I} \sqrt[3]{r_{ki} r_{kj}^2}}{\sqrt{\frac{\epsilon^*}{K}} + r_k} \right)^2$$

Seja $f_{ki} = \frac{\sum u_{k,i}}{\sum_{k=1}^{K} \sum u_{k,i}}$. Assim,

$$n_i = \max_k \left(\frac{\sum u_{k,i}}{f_{ki}} \right) \tag{5.5}$$

5.3 Algoritmo

VFFCM é uma sequência de execuções do algoritmo FCM modificado. Na Figura 2.8, a modificação no FCM consiste em substituir a condição de parada da linha 9 por:

$$\sum_{k=1}^{C} \left(\|c_k - \underline{c}_k\|^2 \right) < \gamma \quad \text{ou} \quad i == I. \tag{5.6}$$

A condição de parada do VFFCM é dado por:

$$\text{Inequação(3.2)} \quad \text{e} \quad \sum_{k=1}^{C} \left(\|c_k - \underline{c}_k\|^2 \right) < \gamma \tag{5.7}$$

De maneira similiar ao VFKM, o número de exemplos usados em todas as iterações da primeira execução do FCM é o mesmo para todas as iterações e é definido usando a fórmula de erro do limite de Hoeffding (colocando n em evidência) mais 10% desse valor:

$$n_i = 1.1 \frac{K}{2} \left(\frac{R}{\epsilon^*} \right)^2 \ln \frac{2}{\delta}. \tag{5.8}$$

Nas execuções seguintes do FCM, o número de exemplos em cada iteração i é determinado pela Equação (5.5) usando a iteração i correspondente da execução anterior do FCM (que teve I iterações). Na primeira execução,

VFFCM define I que é usado para calcular $\delta = 1 - \sqrt[KDI]{1-\delta^*}$. Em (5.6), se i é igual a I e a condição à esquerda não é alcançada, então para a próxima execução do FCM, da mesma forma que o VFKM, I é acrescido de um fator de incremento α.

VFFCM e VFKM são fornecidos juntos no algoritmo da Figura 5.1, em um único pseudo-código. As variáveis de entrada são dadas na Tabela 5.1.

Variável	**Descrição**
K	Número de clusters
T	Conjunto de exemplos $\{x_1, ..., x_N\}$
m	Parâmetro de nebulosidade
γ	Limiar de convergência
δ^* - 1	Prob. VFFCM se aproximar do modelo criado por FCM
ϵ^*	Erro total
I	Número máximo de iterações inicial
α	Fator de incremento de I

Tabela 5.1: Variáveis de Entrada do algoritmo da Figura 5.1

As diferenças entre os algoritmo VFFCM e VFKM resultam por ser apenas o uso do algoritmo FCM ou K-Means, o cálculo do erro ϵ_{kdi} e o número de exemplos n_i usados a cada iteração. Pode ser claramente percebida a semelhança entre o código da Figura 5.1, que apresenta os algoritmos VFFCM e VFKM, e a Figura 4.3, onde o método geral é apresentado. Tanto o VFFCM quanto o VFKM têm basicamente a mesma estrutura, variando apenas no algoritmo base que foi utilizado e nos cálculos de erro e do número de exemplos, exatamente como já havia sido definido no método geral.

Algoritmo Very Fast K/FC-Means

Entrada: Ver Tabela 5.1

Saída: C - Conjunto de centroids $\{c_1, ..., c_K\}$

1: n_i = Equação (5.8) $\quad \forall i, \quad 1 \leq i \leq I$

2: **repita**

3:

VFFCM
Executar FCM

ou

VFKM
Executar K-Means

usando a condição de parada (5.6), com n_i exemplos a cada it. i

4:

VFFCM
ϵ_{kdi} = Equação (5.3)

ou

VFKM
ϵ_{kdi} = Equação (3.3)

5: **bool** *parar* = condição de parada (5.7)

6: **se** *parar*==falso **então**

7: $novoI = I + \alpha I$

8: $n_i = \text{CalcularNi}(I, novoI) \quad \forall i, \quad 1 \leq i \leq novoI$

9: $I = novoI$

10: **fim se**

11: **até** *parar*==verdadeiro

Figura 5.1: Algoritmo Very Fast K/FC-Means

Algoritmo Procedimento CalcularNi()

Entrada: I - Número máximo de iterações utilizado na última execução do algoritmo base

Entrada: $novoI$ - Número máximo de iterações a ser utilizado na próxima execução do algoritmo base

Saída: n_i - Número de exemplos a serem usados a cada iteração i do algoritmo base

1: **para** $i = 1$ até I **faça**

2:

VFFCM	ou	VFKM
$n_i =$ Equação (5.5)		$n_i =$ Equação (3.4)

3: **fim para**

4: **para** $i = I + 1$ até $novoI$ **faça**

5: $n_i =$ Equação (5.8)

6: **fim para**

Figura 5.2: Procedimento CalcularNi()

Capítulo 6

Resultados Experimentais

Em (HULTEN, DOMINGOS, 2003), VFKM foi implementado em linguagem C, enquanto nós implementamos FCM e VFFCM em Java, utilizando os recursos fornecidos pelo conjunto de ferramentas de programação Weka (GARNER, 1995; WITTEN, et al., 1999; WITTEN, FRANK, 2005). Os experimentos foram executados em computadores Pentium 4 1.8GHz, com 1GB RAM, usando o sistema operacional Linux e fazendo uso do mesmo gerador usado em (DOMINGOS, HULTEN, 2001), que funciona gerando bancos de dados através de misturas de Gaussianas esféricas com médias μ_k. Três parâmetros foram usados para gerar os dados: a dimensionalidade D, o número de componentes K e o desvio padrão de cada coordenada σ. As médias μ_k foram geradas uma por vez por amostragem uniforme de cada dimensão dentro do domínio $(2\sigma, 1 - 2\sigma)$. O domínio de cada dimensão foi definido como sendo um. Exemplos x foram gerados escolhendo uma das médias μ_k com probabilidade uniforme, e definido o valor de cada dimensão do exemplo x_d pela amostragem randômica da distribuição Gaussiana com média μ_{kd} e desvio-padrão σ.

A mesma metodologia experimental presente em (FARNSTROM, et al., 2000) foi usada: 30 bancos de dados diferentes foram gerados, cada um com 1 milhão de exemplos, com $D = 100$, com $K = 5$ e $\sigma = 0,1$. Os 30 bancos de dados foram passados 5 vezes cada um, por cada algoritmo, utilizando diferentes inicializações aleatórias dos clusters. Na análise que se segue, serão utilizados como resultados o melhor agrupamento de cada um desses conjuntos de 5 rodadas por cada dataset. A inicialização das variáveis do algoritmo VFKM/VFCM pode ser vista na Tabela 6.1.

Variável	**Valor**
m	$1,5$
γ	$0,0001KD$
δ^*	$0,05$
ϵ^*	$\gamma/3$
I	5 (valor inicial)
α	$0,5$

Tabela 6.1: Inicialização das Variáveis

Os resultados experimentais podem ser vistos nas Tabelas 6.4, 6.5 e 6.6. Nessas três tabelas, a coluna **BD** define o banco de dados (*dataset*), a segunda exibe o algoritmo utilizado, a coluna **Tempo** mostra o tempo gasto por cada algoritmo para o referido dataset, as colunas **PC**, **C/S** e **PBMF** exibem, respectivamente, os cálculos dos índices de validação: Coeficiente de Partição, Compactação/Separação e PBMF, que já foram detalhados na seção 2.3 (os valores referentes ao índice PBMF foram normalizados entre 0 e 1,0). Finalmente, a coluna **Perda** mostra a qualidade do agrupamento de cada algoritmo de acordo com o cálculo da perda comparada com os clusters reais de cada dataset. Cada linha dessas tabelas exibe os resultados da rodada de melhor agrupamento, em cada banco de dados, dentre as 5 rodadas

a que cada um foi exposto em cada algoritmo. Em cada dataset, o melhor valor para cada índice de validação está grifado em negrito.

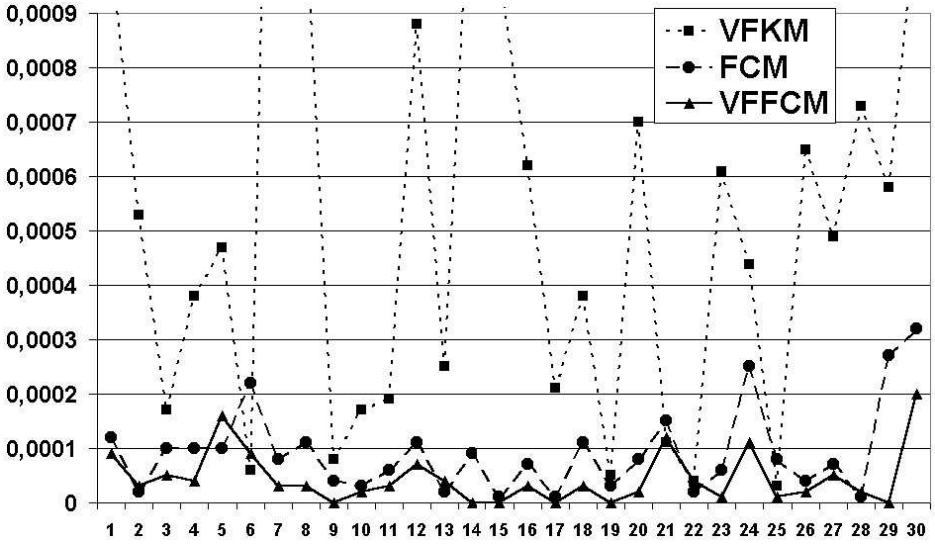

Figura 6.1: Perda em relação aos clusters reais × Dataset

Na Figura 6.1 podemos ver graficamente a comparação entre os valores contidos na coluna **Perda** das Tabelas 6.4, 6.5 e 6.6. Para computar a função de perda (Equação 3.1), comparamos os centroids obtidos com os verdadeiros centroids que geraram os dados. Correlacionamos cada par de centroids na função de perda utilizando um algoritmo guloso, ou seja, dentre os clusters reais e os clusters obtidos com o agrupamento, relacionamos primeiro o par de clusters mais próximos, em seguida o segundo par de clusters mais próximos e assim por diante, até correlacionarmos todos os $K = 5$ pares.

Podemos ver na Figura 6.1 que a curva da qualidade do agrupamento do VFFCM é similar à curva do FCM, o que mostra que os modelos resultantes produzidos por eles realmente não se diferem significativamente. O algoritmo VFFCM obteve uma melhor qualidade no agrupamento em 23 do 30 datasets

(FCM ganhou 5 vezes e VFKM ganhou 2 vezes) mostrando que, na maioria das vezes, o VFFCM obteve melhores resultados. Levando em consideração apenas os algoritmos VFFCM e VFKM, o melhor agrupamento chega a 27 dos 30 datasets apresentados, restando 2 datasets ganhos pelo VFKM e 1 empate (dataset 22).

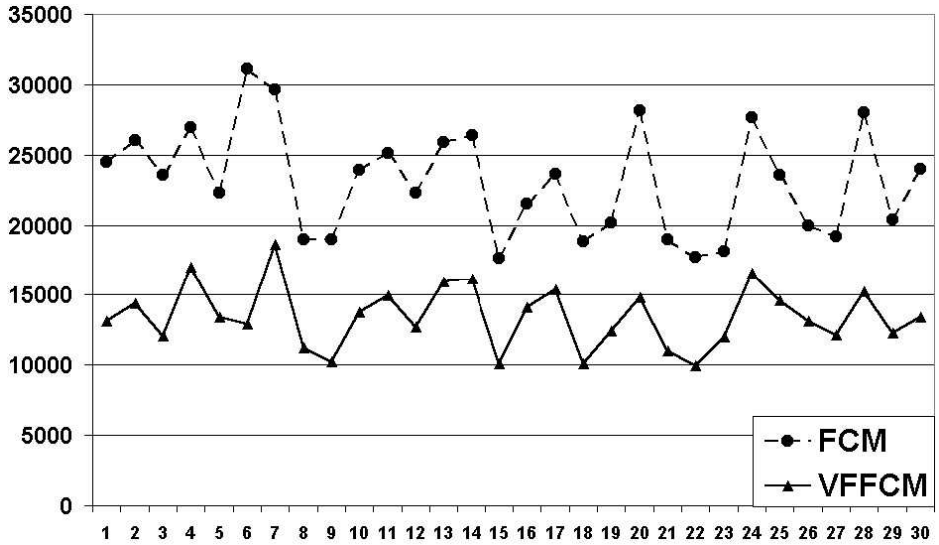

Figura 6.2: Tempo em segundos × Dataset

Na Figura 6.2 mostramos o tempo, em segundos, necessário à execução do banco de dados em cada algoritmo. Nela podemos ver o *speedup* (proporção de aumento de velocidade) do VFFCM relativo ao FCM, pois em todos os datasets o VFFCM foi pelo menos uma vez e meia mais rápido, sendo até $2,4$ vezes mais rápido do que o FCM, no melhor caso. O VFKM não participa dessa comparação pois foi implementado em linguagem C, enquanto que implementamos o FCM e o VFFCM em Java, não sendo possível então terem suas velocidades comparadas, devido à grande diferença de eficiência entre essas duas diferentes linguagens de programação.

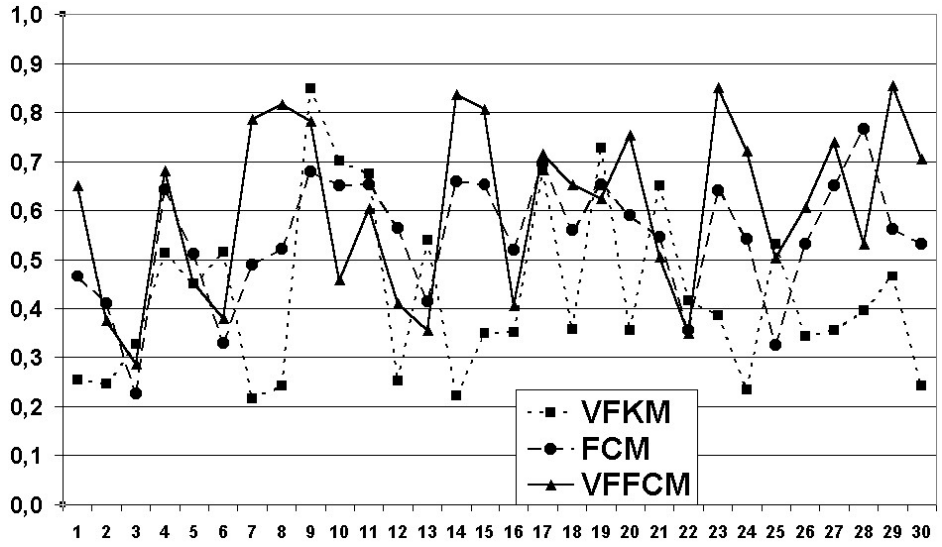

Figura 6.3: Índice de Coeficiente de Partição (máximo para agrupamento crisp)

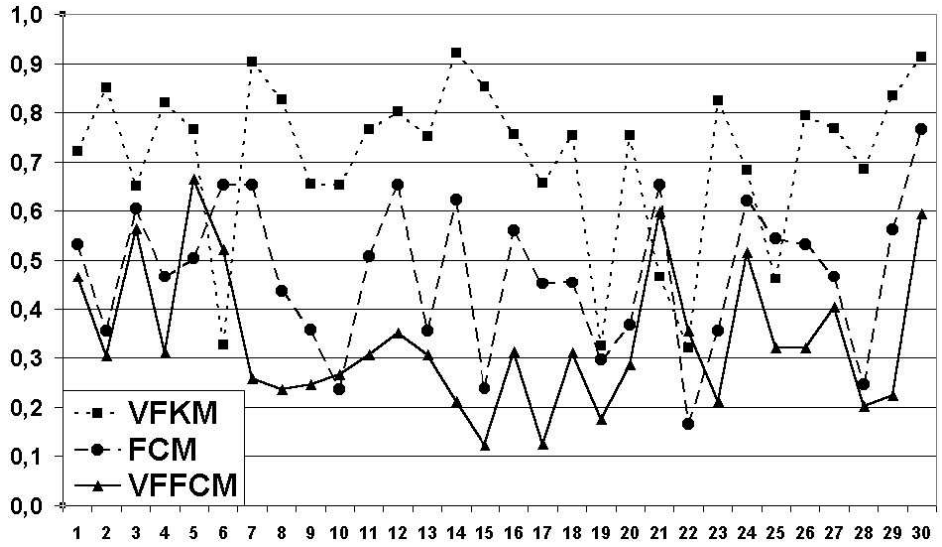

Figura 6.4: Índice de Compatação/Separação (mínimo para clusters compactos e bem separados)

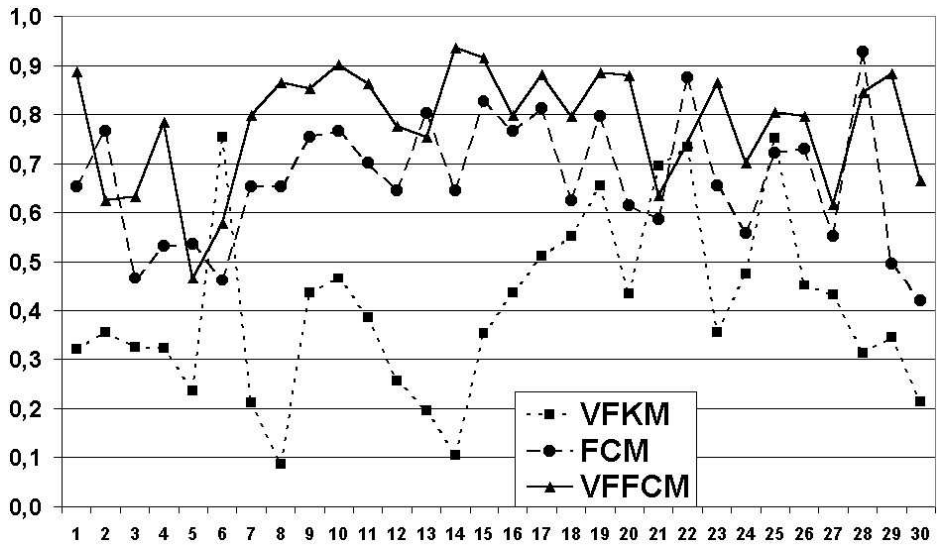

Figura 6.5: Índice PBM(F) (máximo para clusters compactos e bem separados)

Nas Figuras 6.3, 6.4 e 6.5, podemos ver graficamente os dados contidos, respectivamente, nas colunas **PC**, **C/S** e **PBMF** das Tabelas 6.4, 6.5 e 6.6. Por essas três figuras percebemos as diferenças entre cada índice de validação: Na Figura 6.3 as curvas dos três algoritmos estão bem próximas e se confundem. Na Figura 6.4, como acontece na Figura 6.1, a curva do algoritmo VFKM já está relativamente mais separada das demais (apesar de ainda ter vários pontos de interseção, principalmente entre os datasets 19 e 23). Mas somente na Figura 6.5 podemos notar o mesmo comportamento obtido na função perda (Figura 6.1), ou seja, o curva do algoritmo VFKM bastante dispersa, enquanto que as curvas do VFFCM e do FCM seguem um mesmo padrão de comportamento. Para essa comparação é importante levar em consideração que a função perda busca os valores mínimos para o melhor agrupamento, enquanto que o índice PBMF busca os valores máximos. Portanto devemos comparar esses dois gráficos (Figuras 6.5 e 6.1) desta forma.

Algoritmo	**PC**	**C/S**	**PBMF**	**Perda**
VFKM	9	2	2	2
FCM	5	3	5	5
VFFCM	16	25	23	23

Tabela 6.2: Ganho de Cada Algoritmo em Relação à Qualidade do Agrupamento

Índice	**BD**	**Algoritmo indicado pelo índice**	**Algoritmo indicado pela perda**
PC	$3, 9, 10, 11, 19, 25$	VFKM	VFFCM
	$12, 16$	FCM	VFFCM
	$13, 22$	VFKM	FCM
C/S	$2, 13, 28$	VFFCM	FCM
	10	FCM	VFFCM
PBM(F)	-	-	-

Tabela 6.3: Discordâncias Entre os Índices de Validação e a Perda Real

Os dados, portanto, demonstram a melhor qualificação do índice C/S e, principalmente, do índice PBMF em relação ao índice PC, pois eles conseguiram um comportamento mais próximo da perda real do que esse último, como pode ser visto na Tabela 6.2. Podemos notar, através da análise da Tabela 6.3, que a pior qualificação do índice PC se deve ao fato deste índice ter uma tendência a qualificar melhor o algoritmo VFKM, mesmo quando ele não é o melhor. Isto aconteceu principalmente nos datasets em que os valores da função perda do VFKM estão mais próximos das curvas dos outros dois algoritmos fuzzy, levando o índice a um erro de classificação (a saber, isto ocorreu nos bancos de dados $3, 9, 10, 11, 13, 19, 22$ e 25, como visto na Figura 6.1). Creditamos isso ao fato de, como dissemos na seção 2.3.1,

o índice PC fornecer valores máximos para partições crisp, portanto dando preferência ao algoritmo VFKM, por sua característica crisp, quando sua perda se aproxima da perda dos outros dois algoritmos.

O algoritmo VFFCM também foi comparado, em relação à tempo de execução, ao método de amostragem progressiva, descrito na seção 2.4. Utilizando uma amostragem geométrica com tamanho de valor 100 para a amostra inicial ($n_0 = 100$) e duplicando o tamanho da amostra a cada rodada ($a = 2$), como proposto em (PROVOST, et al., 1999), obtivemos os tempos, em segundos, mostrados no gráfico da Figura 6.6.

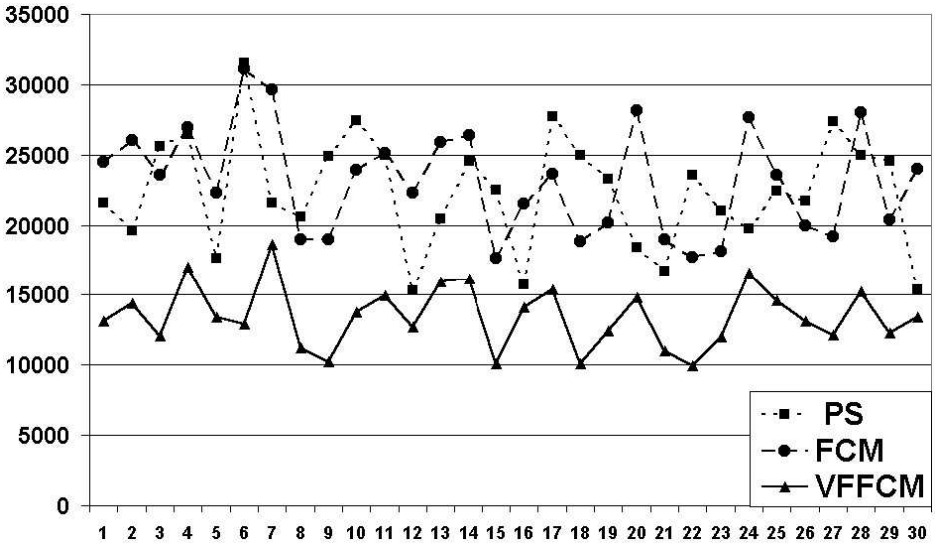

Figura 6.6: Comparação de tempo com a amostragem progressiva - Tempo em segundos × Dataset

É possível observar que os valores da amostragem progressiva não são bons, raramente chegando até mesmo próximos aos tempos do VFFCM, sendo inclusive piores que o próprio algoritmo FCM em alguns casos. Podemos creditar este comportamento à escolha sugerida em (PROVOST, et al.,

1999) do valor da amostra inicial n_0, que funcionou adequadamente para os grandes datasets onde foram aplicados, mas é uma inicialização muito fraca para bases de dados arbitrariamente grandes, que é o caso dos datasets que estamos aplicando.

BD	**Algoritmo**	**Tempo**	**PC** (max)	**C/S** (min)	**PBMF** (max)	**Perda** (min)
1	VFKM		0.25464	0.72155	0.32105	0.00101
	FCM	24512	0.46565	0.53154	0.65324	0.00012
	VFFCM	13112	**0.65161**	**0.46484**	**0.88671**	**0.00009**
2	VFKM		0.24546	0.85122	0.35402	0.00053
	FCM	26007	**0.41026**	0.35401	**0.76542**	**0.00002**
	VFFCM	14372	0.37515	**0.30508**	0.62485	0.00003
3	VFKM		**0.32648**	0.65123	0.32544	0.00017
	FCM	23563	0.22647	0.60554	0.46523	0.00010
	VFFCM	12039	0.28644	**0.56540**	**0.63217**	**0.00005**
4	VFKM		0.51344	0.82009	0.32218	0.00038
	FCM	26935	0.64214	0.46515	0.53187	0.00010
	VFFCM	17017	**0.68123**	**0.31057**	**0.78512**	**0.00004**
5	VFKM		0.44864	0.76542	0.23548	0.00047
	FCM	22306	**0.51289**	**0.50357**	**0.53624**	**0.00010**
	VFFCM	13373	0.45198	0.66580	0.46502	0.00016
6	VFKM		**0.51684**	**0.32568**	**0.75412**	**0.00006**
	FCM	31111	0.32945	0.65238	0.46232	0.00022
	VFFCM	12898	0.37964	0.52177	0.57852	0.00009
7	VFKM		0.21657	0.90240	0.21208	0.00221
	FCM	29615	0.48932	0.65321	0.65301	0.00008
	VFFCM	18653	**0.78615**	**0.25854**	**0.79852**	**0.00003**
8	VFKM		0.24219	0.82624	0.08632	0.00108
	FCM	19012	0.52167	0.43525	0.65421	0.00011
	VFFCM	11233	**0.81642**	**0.23584**	**0.86511**	**0.00003**
9	VFKM		**0.84895**	0.65582	0.43510	0.00008
	FCM	18971	0.68028	0.35685	0.75412	0.00004
	VFFCM	10226	0.78235	**0.24501**	**0.85284**	**0.00000**
10	VFKM		**0.70219**	0.65401	0.46655	0.00017
	FCM	23915	0.65209	**0.23548**	0.76515	0.00003
	VFFCM	13751	0.45682	0.26547	**0.90120**	**0.00002**

Tabela 6.4: Resultados Experimentais (Bancos de Dados de 1 a 10)

BD	**Algoritmo**	**Tempo**	**PC** (max)	**C/S** (min)	**PBMF** (max)	**Perda** (min)
11	VFKM		**0.67477**	0.76545	0.38510	0.00019
	FCM	25142	0.65408	0.50748	0.70129	0.00006
	VFFCM	14967	0.60514	**0.30564**	**0.86214**	**0.00003**
12	VFKM		0.25199	0.80264	0.25641	0.00088
	FCM	22265	**0.56405**	0.65232	0.64421	0.00011
	VFFCM	12712	0.41028	**0.35058**	**0.77655**	**0.00007**
13	VFKM		**0.54010**	0.75213	0.19601	0.00025
	FCM	25913	0.41387	0.35540	**0.80255**	**0.00002**
	VFFCM	15924	0.35407	**0.30623**	0.75421	0.00004
14	VFKM		0.22218	0.92185	0.10523	0.00124
	FCM	26425	0.66021	0.62321	0.64551	0.00009
	VFFCM	16187	**0.83613**	**0.21254**	**0.93507**	**0.00000**
15	VFKM		0.34805	0.85340	0.35212	0.00098
	FCM	17620	0.65320	0.23854	0.82642	0.00001
	VFFCM	10098	**0.80657**	**0.12365**	**0.91544**	**0.00000**
16	VFKM		0.35105	0.75641	0.43508	0.00062
	FCM	21524	**0.52100**	0.56005	0.76515	0.00007
	VFFCM	14138	0.40547	**0.31201**	**0.79852**	**0.00003**
17	VFKM		0.68240	0.65654	0.51112	0.00021
	FCM	23650	0.69523	0.45211	0.81317	0.00001
	VFFCM	15359	**0.71567**	**0.12504**	**0.88071**	**0.00000**
18	VFKM		0.35654	0.75321	0.55321	0.00038
	FCM	18866	0.56087	0.45370	0.62453	0.00011
	VFFCM	10125	**0.65270**	**0.31025**	**0.79541**	**0.00003**
19	VFKM		**0.72870**	0.32555	0.65512	0.00005
	FCM	20199	0.65273	0.29560	0.79562	0.00003
	VFFCM	12386	0.62487	**0.17565**	**0.88568**	**0.00000**
20	VFKM		0.35560	0.75455	0.43257	0.00070
	FCM	28142	0.58972	0.36666	0.61580	0.00008
	VFFCM	14853	**0.75402**	**0.28547**	**0.87805**	**0.00002**

Tabela 6.5: Resultados Experimentais (Bancos de Dados de 11 a 20)

BD	**Algoritmo**	**Tempo**	**PC** (max)	**C/S** (min)	**PBMF** (max)	**Perda** (min)
21	VFKM		**0.65211**	**0.46532**	**0.69652**	**0.00011**
	FCM	19007	0.54568	0.65421	0.58617	0.00015
	VFFCM	10997	0.50544	0.59874	0.63504	0.00012
22	VFKM		**0.41597**	0.32152	0.73456	0.00004
	FCM	17719	0.35480	**0.16560**	**0.87409**	**0.00002**
	VFFCM	9982	0.34874	0.35408	0.74125	0.00004
23	VFKM		0.38544	0.82404	0.35444	0.00061
	FCM	18113	0.64064	0.35477	0.65482	0.00006
	VFFCM	11971	**0.85140**	**0.21114**	**0.86547**	**0.00001**
24	VFKM		0.23477	0.68421	0.47545	0.00044
	FCM	27671	0.54242	0.62155	0.55821	0.00025
	VFFCM	16564	**0.72150**	**0.51584**	**0.70108**	**0.00011**
25	VFKM		**0.53172**	0.46221	0.75214	0.00003
	FCM	23547	0.32548	0.54521	0.72215	0.00008
	VFFCM	14641	0.50450	**0.32150**	**0.80520**	**0.00001**
26	VFKM		0.34284	0.79521	0.45214	0.00065
	FCM	19962	0.53242	0.53214	0.73022	0.00004
	VFFCM	13137	**0.60640**	**0.32155**	**0.79621**	**0.00002**
27	VFKM		0.35420	0.76899	0.43215	0.00049
	FCM	19198	0.65221	0.46580	0.55211	0.00007
	VFFCM	12112	**0.74012**	**0.40235**	**0.61658**	**0.00005**
28	VFKM		0.39521	0.68532	0.31207	0.00073
	FCM	28030	**0.76523**	0.24652	**0.92654**	**0.00001**
	VFFCM	15234	0.53240	**0.20158**	0.84522	0.00002
29	VFKM		0.46552	0.83514	0.34486	0.00058
	FCM	20389	0.56210	0.56231	0.49521	0.00027
	VFFCM	12271	**0.85474**	**0.22359**	**0.88221**	**0.00000**
30	VFKM		0.24219	0.91237	0.21354	0.00107
	FCM	23987	0.53210	0.76521	0.42007	0.00032
	VFFCM	13383	**0.70521**	**0.59563**	**0.66441**	**0.00020**

Tabela 6.6: Resultados Experimentais (Bancos de Dados de 21 a 30)

Capítulo 7

Conclusão

Como apontado em (HAMERLY, ELKAN, 2002), a utilização da função de pertinência fuzzy é essencial para obter bons agrupamentos. Portanto, adaptamos o método de (DOMINGOS, HULTEN, 2001) para tornar algoritmos de aprendizado de máquinas escaláveis para bases de dados arbitrariamente grandes e o aplicamos ao FCM, desenvolvendo o algoritmo Very Fast Fuzzy C-Means (VFFCM). Em (PAL, BEZDEK, 2002), esta adaptação foi considerada uma importante questão ainda em aberto, por não ser um problema de solução simples e direta.

Nós comparamos VFFCM com o VFKM e o FCM, em 30 diferentes conjuntos de dados (com 1 milhão de exemplos e 100 dimensões cada), utilizando como índice de qualidade do agrupamento: a perda relativa aos verdadeiros clusters e três diferentes índices de qualidade de agrupamento.

VFFCM alcançou seus objetivos principais: os resultados da qualidade de agrupamento do VFFCM são pelo menos tão bons quanto os resultados obtidos passando todos os dados pelo FCM, o que mostra que o modelo produzido pelo VFFCM realmente não difere significantemente daquele produzido pelo FCM. Além disso, foi observado um speedup do VFFCM relativo ao FCM,

onde o VFFCM foi pelo menos $1,5$ vezes mais eficiente, sendo até $2,4$ vezes mais eficiente no melhor caso. O algoritmo VFKM obteve um speedup similar em relação ao K-Means. Os resultados experimentais também mostraram que a qualidade de agrupamento do VFFCM é quase sempre melhor do que a do VFKM (27 de 30), como esperado já que FCM costuma fornecer melhores agrupamentos do que o K-Means.

Outra importante contribuição desse trabalho é o algoritmo da Figura 5.1, onde é apresentado não apenas o pseudo-código do novo algoritmo VFFCM, mas também é mostrado de forma simples o pseudo-código referente ao algoritmo VFKM, que não havia sido apresentado de forma tão clara em (DOMINGOS, HULTEN, 2001). Apresentar o algoritmo dessa maneira é muito importante para a melhor visualização de seu funcionamento, além disso facilita a diferenciação entre os algoritmos VFKM e VFFCM.

O método descrito em (DOMINGOS, HULTEN, 2001) foi relacionado a outros importantes trabalhos na literatura (MARON, MOORE, 1994; JOHN, LANGLEY, 1996), incluindo algoritmos de agrupamento escaláveis e amostragem progressiva (PROVOST, et al., 1999; BEZDEK, HATHAWAY, 2004). A abordagem da amostragem progressiva é desfavorecida pelo fato das curvas reais de aprendizado não acompanharem leis poderosas e outras formas simples boas o suficiente para uma extrapolação segura (PROVOST, et al., 1999). Como apontado em (DOMINGOS, HULTEN, 2001), a desvantagem de seu método comparado com o a amostragem progressiva é que mais exemplos que o necessário talvez sejam usados para se alcançar a garantia do método, enquanto que sua vantagem é que ele permite optimizar o número de exemplos a cada passo do aprendizado, em oposição a determinar o número de exemplos usado por todo o algoritmo. Essas características provenientes

do método utilizado no algoritmo VFKM, portanto, podem ser extendidas ao VFFCM.

Comparando com o agrupamento no ambiente de aprendizado teórico PAC (VALIANT, 1984; HAUSSLER, 1988; HAUSSLER, 1992; MISHRA, et al., 2001), a inovação do método existente em (DOMINGOS, HULTEN, 2001) é que os limites são para a diferença entre o modelo aprendido com finitos e infinitos dados para o mesmo algoritmo, e não para o melhor modelo da classe. Desta forma, acreditamos que esses limites são mais precisos e evitam aprendizado desnecessário. Os limites usados pelo PAC costumam precisar de mais dados e não fornecem speedup em bases de dados extremamente grandes.

Como trabalho futuro, esse método pode ser aplicado em outros algoritmos soft e algoritmos de aprendizado de máquinas, como redes neurais. Essas aplicações forneceriam informações importantes sobre o funcionamento e adaptação do método aos diversos tipos de algoritmos existentes no campo da Mineração de Dados.

Outro trabalho futuro seria a comparação do algoritmo VFFCM com a amostragem progressiva de uma forma diferentes da maneira como foi abordada neste trabalho. Para obter os resultados experimentais, foi utilizada para a amostra inicial um valor fixo de tamanho 100, como sugerido em (PROVOST, et al., 1999). Seria interessante utilizar não apenas uma constante para o tamanho da amostra inicial, e sim uma abordagem eficiente, como em (GU, et al., 2001), para melhorar a amostragem progressiva, já que da forma que ela foi utilizada acabou mostrando, algumas vezes, resultados piores até mesmo em relação ao algoritmo original.

Um último trabalho futuro seria a utilização de versões mais avançadas do FCM, como uma das diversas versões descritas na seção 2.2. Por ser um

algoritmo geral, o FCM acaba sendo mais conservador na eficiência e nos modelos produzidos. Porém para aplicação do método em algum problema específico, a utilização do método em um algoritmo especialmente desenvolvido para este fim tornaria o algoritmo ainda mais eficiente.

Referências Bibliográficas

AGGARWAL, C. C., WOLF, J. L., YU, P. S., PROCOPIUC, C., PARK, J. S., 1999, "Fast algorithms for projected clustering", In:Delis, A., Faloutsos, C., Ghandeharizadeh, S., , editors, *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data: SIGMOD '99, Philadelphia, PA, USA, June 1–3, 1999*, volume 28(2) of *SIGMOD Record (ACM Special Interest Group on Management of Data)*, pp. 61–72, New York, NY 10036, USA ACM Press.

AGRAWAL, R., GEHRKE, J., GUNOPULOS, D., RAGHAVAN, P., 1998, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications", *j-SIGMOD*, v. 27, n. 2, pp. 94–105.

AHMED, M. N., YAMANY, S. M., MOHAMED, N., FARAG, A. A., MORIARTY, T., 2002, "A modified fuzzy c-means algorithm for bias field estimation and segmentation of MRI data", *IEEE Transactions on Medical Imaging*, v. 21, n. 3, pp. 193–199.

ALEXIUK, M. D., PIZZI, N. J., 2005, "Robust centroids using fuzzy clustering with feature partitions", *Pattern Recognition Letters*, v. 26, n. 8, pp. 1039–1046.

ASHARAF, S., MURTY, M., 2003, "An adaptive rough fuzzy single pass algorithm for clustering large data sets", *Pattern Recognition*, v. 36, n. 12, pp. 3015–3018.

BANFIELD, J. D., RAFTERY, A. E., 1993, "Model-Based Gaussian and Non-Gaussian Clustering", *Biometrics*, v. 49, pp. 803–821.

BARBIERI, P., ADAMI, G., FAVRETTO, A., LUTMAN, A., REISENHOFER, E., 2001, "Robust cluster analysis for detecting physico-chemical typologies of freshwater from wells of the Plain of Friuli (north-eastern Italy)", *Analytica Chimica Acta*, v. 440, pp. 161–169.

BELACEL, N., HANSEN, P., MLADENOVIC, N., 2002, "Fuzzy J-Means: a new heuristic for fuzzy clustering", *Pattern Recognition*, v. 35, n. 10, pp. 2193–2200.

BEZDEK, J. C., 1981, *Pattern recognition with fuzzy objective function algorithms*, Plenum Press, New York.

BEZDEK, J. C., EHRLICH, R., FULL, W., 1984, "FCM: The fuzzy c–means clustering algorithm", *Computers and Geosciences*, v. 10, n. 2–3, pp. 191–203.

BEZDEK, J. C., HALL, L., CLARKE, L., 1993, "Review of MR image segmentation techniques using pattern recognition", *Medical Physics*, v. 20, n. 4, pp. 1033–1048.

BEZDEK, J. C., HATHAWAY, R. J., 2004, "Progressive sampling schemes for approximate clustering in very large data sets", *IEEE International Conference on Fuzzy Systems*, v. 1, pp. 15–21.

BEZDEK, J. C., PAL, N. R., KELLER, J., KRISNAPURAM, R., 1999, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*, Kluwer Academic Publishers.

BISHOP, C. M., 1995, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford.

BRACHMAN, R. J., KHABAZA, T., KLOESGEN, W., PIATETSKY-SHAPIRO, G., SIMOUDIS, E., 1996, "Mining Business Databases", *j-CACM*, v. 39, n. 11, pp. 42–48.

BRADLEY, P. S., FAYYAD, U. M., 1998, "Refining initial points for K-Means clustering", In:*Proceedings 15th International Conference on Machine Learning*, pp. 91–99 Morgan Kaufmann, San Francisco, CA.

BRADLEY, P. S., FAYYAD, U. M., REINA, C., 1998, "Scaling Clustering Algorithms to Large Databases", In:*Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 9–15 AAAI Press.

BRADLEY, P. S., MANGASARIAN, O. L., STREET, W. N., 1996, "Clustering via Concave Minimization", In:*NIPS*, pp. 368–374.

Brown, C., , editor1988, *Advances in Computer Vision*, Lawrence Erlbaum Associates, New York.

BURROUGH, P. A., GAANS, P. F. M. V., MACMILLAN, R. A., 2000, "High-resolution landform classification using fuzzy k-means", *Fuzzy Sets Syst.*, v. 113, n. 1, pp. 37–52.

CANNON, R. L., DAVE, J. V., BEZDEK, J. C., 1986, "Efficient implementation of the fuzzy c-means clustering algorithms", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, v. 8, n. 2, pp. 248–255.

CATLETT, J., 1991a, "Megainduction: A Test Flight", In:*ML*, pp. 596–599.

CATLETT, J., 1991b, *Megainduction: machine learning on very large databases*, Ph.D. thesis, School of Computer Science, University of Technology, Sydney, Australia.

CORDEIRO, T., ZAVERUCHA, G., 2005, "Tornando Fuzzy C-Means Escalável para Bancos de Dados Arbitrariamente Grandes", In: *V Encontro Nacional de Inteligência Artificial (ENIA'2005)*, pp. 742–751, São Leopoldo, Rio Grande do Sul, Brasil.

CORDEIRO, T., ZAVERUCHA, G., 2006, "Um Método Geral para Tornar Algoritmos Fuzzy de Aprendizado de Máquinas Escaláveis para Bases de Dados Arbitrariamente Grandes", In:*XVI Congresso Brasileiro de Automática (CBA 2006)*, Paper aceito.

DAVIES, D., BOULDIN, D., 1979, "A cluster separation measure", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 1, n. 2, pp. 224–227.

DIDAY, E., SIMON, J. C., 1976, "Clustering Analysis", In:Fu, K. S., , editor, *Digital Pattern Recognition*, pp. 47–94 Springer Verlag.

DOMINGOS, P., HULTEN, G., 2000, "Mining high-speed data streams", In:*KDD'00: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 71–80, New York, NY, USA ACM Press.

DOMINGOS, P., HULTEN, G., 2001, "A General Method for Scaling Up Machine Learning Algorithms and its Application to Clustering", In:*ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 106–113, San Francisco, CA, USA Morgan Kaufmann Publishers Inc.

DOMINGOS, P., HULTEN, G., 2002, "Learning from Infinite Data in Finite Time", In:*Advances in Neural Information Processing Systems 14*, Cambridge, MA MIT Press.

DOMINGOS, P., HULTEN, G., 2003, "A General Framework for Mining Massive Data Streams", *Journal of Computational and Graphical Statistics*, v. 12, n. 4, pp. 945–949.

DUDA, R. O., HART, P. E., 1973, *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York.

DUNN, J., 1973, "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters", *Journal of Cybernetics*, v. 3, pp. 32–57.

EL-SONBATY, Y., ISMAIL, M. A., 1998, "Fuzzy Clustering for Symbolic Data", *IEEE Transactions on Fuzzy Systems*, v. 6, n. 2, pp. 195–204.

ESTER, M., KRIEGEL, H.-P., SANDER, J., XU, X., 1996, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", In:Simoudis, E., Han, J. W., Fayyad, U., , editors, *KDD'96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, Portland, Oregon AAAI Press.

ESTER, M., KRIEGEL, H.-P., XU, X., 1995, "A Database Interface for Clustering in Large Spatial Databases.", In:*KDD'95: Proceedings of First In-*

ternational Conference on Knowledge Discovery and Data Mining, pp. 94–99.

EVERITT, B., 1974, *Cluster Analysis*,, number 11 in Social Science Research Council Reviews of Current Research Heinemann Educational Books, London.

FAN, J. L., ZHEN, W. Z., XIE, W. X., 2003, "Suppressed fuzzy c-means clustering algorithm", *Pattern Recognition Letters*, v. 24, n. 9-10, pp. 1607–1612.

FARNSTROM, F., LEWIS, J., 2000, "Fast, single-pass K-means algorithms".

FARNSTROM, F., LEWIS, J., ELKAN, C., 2000, "Scalability for Clustering Algorithms Revisited", *SIGKDD Explorations*, v. 2, n. 1, pp. 51–57.

FAYYAD, U., HAUSSLER, D., STOLORZ, P., 1996a, "Mining Scientific Data", *Communications of the ACM*, v. 39, n. 11, pp. 51–57.

FAYYAD, U. M., 1998, "Mining Databases: Towards Algorithms for Knowledge Discovery", *IEEE Data Eng. Bull*, v. 21, n. 1, pp. 39–48.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., , editors1996b, *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press.

FAYYAD, U. M., REINA, C., BRADLEY, P. S., 1998, "Initialization of Iterative Refinement Clustering Algorithms", In:*KDD'98: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 194–198.

FISHER, D. H., 1987, "Knowledge Acquisition Via Incremental Conceptual Clustering", *Machine Learning*, v. 2, pp. 139–172.

FORGY, E. W., 1965, "Cluster analysis of multivariate data: efficiency vs interpretability of classifications", *Biometrics*, v. 21, pp. 768–769.

FRAWLEY, W. J., PIATETSKY-SHAPIRO, G., MATHEUS, C. J., 1991, "Knowledge discovery in databases: an overview", In:Piatetsky-Shapiro, G., Frawley, W. J., , editors, *Knowledge discovery in databases*, pp. 1–27, Menlo Park, CA/Cambridge, MA AAAI Press/MIT Press.

FRIEDMAN, J. H., 1997, "Data mining and statistics: What's the connection?", In:*Proceedings of the 29th Symposium on the Interface Between Computer Science and Statistics*.

FUKUNAGA, K., 1990, *Introduction to Statistical Pattern Recognition*, Academic Press, Inc.

GANTI, V., GEHRKE, J., RAMAKRISHNAN, R., 1999, "Mining Very Large Databases", *IEEE Computer*, v. 32, n. 8, pp. 38–45.

GARNER, S. R., 1995, "WEKA: The Waikato Environment for Knowledge Analysis".

GEVA, A., 1999, "Hierarchical Unsupervised Fuzzy Clustering", *IEEE Transactions on Fuzzy Systems*, v. 7, n. 6, pp. 723–733.

GONZALEZ, R. C., WINTZ, P., 1987, *Digital Image Processing*, second ed., Addison-Wesley, Reading, MA.

GU, B., LIU, B., HU, F., LIU, H., 2001, "Efficiently Determine the Starting Sample Size for Progressive Sampling", In:*DMKD*.

GUHA, S., RASTOGI, R., SHIM, K., 1998, "CURE: an efficient clustering algorithm for large databases", In:Haas, L., Tiwary, A., , editors, *Pro-*

ceedings of the 1998 ACM SIGMOD International Conference on Management of Data: June 1–4, 1998, Seattle, Washington, USA, volume 27(2) of *SIGMOD Record (ACM Special Interest Group on Management of Data)*, pp. 73–84, New York, NY 10036, USA ACM Press.

GUHA, S., RASTOGI, R., SHIM, K., 1999, "ROCK: A Robust Clustering Algorithm for Categorical Attributes", In:*15th International Conference on Data Engineering (ICDE '99)*, pp. 512–521, Washington - Brussels - Tokyo IEEE.

GüLER, C., 2002, *Hydrogeochemical evaluation of the groundwater resources of Indian Wells-Owens Valley area, Southeastern California*, Ph.D. thesis, Colorado School of Mines.

GüLER, C., THYNE, G. D., 2004, "Delineation of hydrochemical facies distribution in a regional groundwater system by means of fuzzy c-means clustering", *Water Resourses Research*, v. 40, , W12503, doi:10.1029/2004WR003299.

HALKIDI, M., BATISTAKIS, Y., VAZIRGIANNIS, M., 2002a, "Cluster validity methods: part I", *j-SIGMOD*, v. 31, n. 2, pp. 40–45.

HALKIDI, M., BATISTAKIS, Y., VAZIRGIANNIS, M., 2002b, "Clustering validity checking methods: part II", *j-SIGMOD*, v. 31, n. 3, pp. 19–27.

HAMERLY, G., ELKAN, C., 2002, "Alternatives to the k-means algorithm that find better clusterings", In:*CIKM '02: Proceedings of the eleventh ACM International Conference on Information and Knowledge Management*, pp. 600–607, New York, NY, USA ACM Press.

HAN, J., KAMBER, M., 2001, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, San Francisco, CA.

HATHAWAY, R. J., BEZDEK, J. C., 2006, "Extending Fuzzy and Probabilistic Clustering to Very Large Data Sets", *Journal of Computational Statistics and Data Analysis*, , In press.

HAUSSLER, D., 1988, "Quantifying inductive bias : AI learning algorithms and Valiant's learning framework", *Artificial Intelligence, an International Journal, Sept. 1988*, v. 36, n. 2.

HAUSSLER, D., 1990, "Decision Theoretic Generalizations of the PAC Learning Model", In:*ALT*, pp. 21–41.

HAUSSLER, D., 1992, "Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications", *Information and Computation*, v. 100, n. 1, pp. 78–150.

HERSHFINKEL, D., DINSTEIN, I., 1996, "Accelerated Fuzzy C-Means Clustering Algorithm", In:Bosachi, B., Bezdek, J. C., , editors, *Proceedings SPIE Applications of Fuzzy Logic Technology III*, volume 2761, pp. 41–52.

HINNEBURG, A., KEIM, D. A., 1998, "An Efficient Approach to Clustering in Large Multimedia Databases with Noise", In:*KDD'98: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 58–65.

HOEFFDING, W., 1963, "Probability inequalities for sums of bounded random variables", *Journal of the American Statistical Association*, v. 58, pp. 13–30.

HU, B. G., GOSINE, R., CAO, L., 1998, "Application of Fuzzy Classification Technique in Computer Grading of Fish Product", *IEEE Transactions on Fuzzy Systems*, v. 6, n. 1, pp. 144–152.

HUBER, P., 1996, "Massive Data Sets Workshop: The Morning After", *National Academy Press*, pp. 169–184.

HULTEN, G., DOMINGOS, P., 2002, "Mining complex models from arbitrarily large databases in constant time", In:*KDD'02: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 525–531, New York, NY, USA ACM Press.

HULTEN, G., DOMINGOS, P., 2003, "VFML – A toolkit for mining high-speed time-changing data streams", http://www.cs.washington.edu/dm/vfml/.

HULTEN, G., DOMINGOS, P., ABE, Y., 2003, "Mining Massive Relational Databases", In:*IJCAI-2003 Workshop on Learning Statistical Models from Relational Data*, pp. 53–60.

HULTEN, G., SPENCER, L., DOMINGOS, P., 2001, "Mining time-changing data streams", In:*KDD'01: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 97–106, New York, NY, USA ACM Press.

HUNG, W. L., YANG, M. S., CHEN, D. H., 2006, "Parameter selection for suppressed fuzzy c-means with an application to MRI segmentation", *Pattern Recognition Letters*, v. 27, n. 5, pp. 424–438.

JAIN, A. K., 1989, *Fundamentals of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, NJ.

JAIN, A. K., DUBES, R. C., 1988, *Algorithms for clustering data*, Prentice Hall International, Englewood Cliffs, NJ.

JAIN, A. K., LAW, M. H. C., 2005, "Data Clustering: A User's Dilemma", In:*Pattern Recognition and Machine Intelligence, First International Conference, PReMI 2005, Kolkata, India, December 20-22, 2005, Proceedings*, volume 3776, pp. 1–10 Springer.

JAIN, A. K., MURTY, M. N., FLYNN, P. J., 1999, "Data clustering: a review", *ACM Comput. Surv.*, v. 31, n. 3, pp. 264–323.

JOHN, G. H., LANGLEY, P., 1996, "Static Versus Dynamic Sampling for Data Mining", In:Simoudis, E., Han, J., Fayyad, U. M., , editors, *Proceedings 2nd International Conference Knowledge Discovery and Data Mining*, pp. 367–370 AAAI Press.

KAUFMAN, L., ROUSSEEUW, P. J., 1990, *Finding Groups in Data: An Introduction to Cluster Analysis.*, John Wiley.

KERSTEN, P., 1999, "Fuzzy order statistics and their application to fuzzy clustering", *IEEE Transactions on Fuzzy Systems*, v. 7, pp. 708–712.

LINUSSON, A., WOLD, S., NORDEN, B., 1998, "Fuzzy clustering of 627 alcohols, guided by a strategy for cluster analysis of chemical compounds for combinatorial chemistry", *Chemometrics and Intelligent Laboratory Systems*, v. 44, n. 1-2, pp. 213–227.

MACQUEEN, J., 1967, "Some methods for classification and analysis of multivariate observations", In:*Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297.

MARON, O., MOORE, A. W., 1994, "Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation",

In:Cowan, J. D., Tesauro, G., Alspector, J., , editors, *Advances in Neural Information Processing Systems*, volume 6, pp. 59–66 Morgan Kaufmann Publishers, Inc.

MCBRATNEY, A., MOORE, A., 1985, "Application of fuzzy sets to climatic classification", *Agricultural and Forest Meteorology*, v. 35, pp. 165–185.

MCBRATNEY, A. B., DEGRUIJTER, J., 1992, "A continuum approach to soil classification by modified fuzzy k-means with extragrades", *Journal of Soil Science*, v. 43, pp. 159–175.

MCNEILL, D., FREIBERGER, P., 1993, *Fuzzy Logic*, Simon & Schuster, New York, NY, USA.

MEILA, M., HECKERMAN, D., 1998, "An Experimental Comparison of Several Clustering and Initialization Methods", In:Cooper, G. F., Moral, S., , editors, *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, July 24-26, 1998, University of Wisconsin Business School, Madison, Wisconsin, USA*, pp. 386–395 Morgan Kaufmann.

MISHRA, N., OBLINGER, D., PITT, L., 2001, "Sublinear time approximate clustering", In:*SODA'01: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 439–447, New York ACM Press.

MITCHELL, T. M., 1997, *Machine Learning*, McGraw-Hill, New York.

NG, R. T., HAN, J., 1994, "Efficient and Effective Clustering Methods for Spatial Data Mining", In:*Proceedings of the Twentieth International Conference on Very Large Databases*, pp. 144–155, Santiago, Chile.

ODEH, I., MCBRATNEY, A., CHITTLEBOROUGH, D., 1992, "Soil pattern recognition with fuzzy c-means: Application to classification and soil-landform interrealtionship", *Soil Science Society of American Journal*, v. 56, pp. 505–516.

PAKHIRA, M. K., BANDYOPADHYAY, S., MAULIK, U., 2004, "Validity index for crisp and fuzzy clusters.", *Pattern Recognition*, v. 37, n. 3, pp. 487–501.

PAL, N. R., BEZDEK, J. C., 2002, "Complexity reduction for "large image" processing", *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, v. 32, n. 5, pp. 598–611.

PEDRYCZ, W., 1996, "Conditional Fuzzy C-Means.", *Pattern Recognition Letters*, v. 17, n. 6, pp. 625–631.

PELLEG, D., MOORE, A., 2000, "extending k-means with efficient estimation of the number of clusters", In:*Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 727–734, San Francisco, CA Morgan Kaufmann.

PENA, J. M., LOZANO, J. A., LARRANAGA, P., 1999, "An empirical comparison of four initialization methods for the K-Means algorithm", *Pattern Recognition Letters*, v. 20, n. 10, pp. 1027–1040.

Piatetsky-Shapiro, G., Frawley, W. J., , editors1991, *Knowledge Discovery in Databases*, AAAI/MIT Press.

PROVOST, F., JENSEN, D., OATES, T., 1999, "Efficient progressive sampling", In:*KDD'99: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 23–32, New York, NY, USA ACM Press.

PROVOST, F. J., KOLLURI, V., 1999, "A Survey of Methods for Scaling Up Inductive Algorithms", *Data Mining and Knowledge Discovery*, v. 3, n. 2, pp. 131–169.

RANTITSCH, G., 2000, "Application of fuzzy clusters to quantify lithological background concentrations in stream-sediment geochemistry", *Journal of Geochemical Exploration*, v. 71, pp. 73–82.

SCOTT, D. W., 1992, *Multivariate Density Estimation*, John Wiley and sons, New York, NY.

SELIM, S. Z., ISMAIL, M. A., 1984, "K–Means type algorithms: A generalized convergence theorem and characterization of local optimality", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 6, pp. 81–86.

SHANKAR, B. U., PAL., N., 1994, "FFCM: An effective approach for large data sets", In:*Proceedings of the 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing*, pp. 331–332, Iizuka, Japan.

SHEIKHOLESLAMI, G., CHATTERJEE, S., ZHANG, A., 1998, "WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases", In:Gupta, A., Shmueli, O., Widom, J., , editors, *Proceedings of the Twenty-fourth International Conference on Very Large Databases, New York, NY, USA, 24–27 August, 1998*, pp. 428–439, Los Altos, CA 94022, USA Morgan Kaufmann Publishers.

SILVERMAN, B. W., 1986, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, New York.

SUN, L.-X., DANZER, K., 1996, "Fuzzy cluster analysis by simulated annealing", *Journal of Chemometrics*, v. 10, pp. 325–342.

THEODORIDIS, S., KOUTROUMBAS, K., 1999, *Pattern Recognition*, first ed., Academic Press, USA.

VALIANT, L. G., 1984, "A Theory of the Learnable", *Communications of the ACM*, v. 27, n. 11, pp. 1134–1142.

WANG, W., YANG, J., MUNTZ, R. R., 1997, "STING: A Statistical Information Grid Approach to Spatial Data Mining", In:Jarke, M., Carey, M. J., Dittrich, K. R., Lochovsky, F. H., Loucopoulos, P., Jeusfeld, M. A., , editors, *Twenty-Third International Conference on Very Large Data Bases*, pp. 186–195, Athens, Greece Morgan Kaufmann.

WANG, W., YANG, J., MUNTZ, R. R., 1999, "STING+: An approach to active spatial data mining", In:*Fifteenth International Conference on Data Engineering*, pp. 116–125, Sydney, Australia IEEE Computer Society.

Winston, P. H., , editor1975, *The Psychology of Computer Vision*, McGraw-Hill, New York.

WITTEN, I., FRANK, E., TRIGG, L., HALL, M., HOLMES, G., CUNNINGHAM, S. J., 1999, "Weka: Practical machine learning tools and techniques with java implementations", In:*Proceedings of ICONIP/ANZIIS/ANNES'99 International Workshop: Emerging Knowledge Engineering and Connectionist-Based Info. Systems.*, pp. 192–196.

WITTEN, I. H., FRANK, E., 2005, *Data Mining: Practical Machine Learning Tools and Techniques*, second ed., Morgan Kaufmann, San Francisco.

WU, K.-L., YANG, M.-S., 2002, "Alternative c-means clustering algorithms", *Pattern Recognition*, v. 35, n. 10, pp. 2267–2278.

XIE, X. L., BENI, G., 1991, "A Validity Measure for Fuzzy Clustering.", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 13, n. 8, pp. 841–847.

ZADEH, L., 1965, "Fuzzy Sets", *Information and Control*, v. 3, n. 8, pp. 338–353.

ZHANG, C., CHOU, K.C.AND MAGGIORA, G., 1995, "Predicting protein structural classes from amino acid composition: application of fuzzy clustering", *Protein Engineering*, v. 8, pp. 425–435.

ZHANG, T., RAMAKRISHNAN, R., LIVNY, M., 1996, "BIRCH: An Efficient Data Clustering Method for Very Large Databases", In:*SIGMOD Conference*, pp. 103–114.