Conversão Simbólica de Sinais Digitais por meio da Teoria de Extremos Relativos

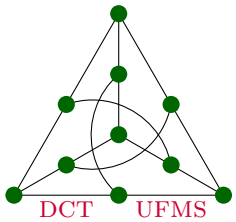
Daniel Joaquim de Sousa

Dissertação de Mestrado

Orientação: Profª. Drª. Maria Bernadete Zanusso

Área de Concentração: Inteligência Artificial

Dissertação apresentada como requisito para obtenção do título de mestre em Ciência da Computação.



Departamento de Computação e Estatística
Centro de Ciências Exatas e Tecnologia
Universidade Federal de Mato Grosso do Sul
9 de julho de 2009

Agradecimentos

A Deus. Dele procede a perfeita sabedoria, conhecimento, riquezas, e bênçãos sem fim.

Aos meus pais, pela educação, princípios e conselhos.

À Professora Doutora Maria Bernadete Zanusso, por sua orientação e paciência.

À minha amada esposa, por seu apoio e compreensão.

Aos meus filhos, nas horas em que tive que lhes privar de minha companhia.

Conteúdo

Lista de Figuras **v**

Lista de Tabelas **vi**

Lista de Siglas **viii**

Resumo **ix**

Abstract **x**

1 Introdução **1**
1.1 Justificativa e Objetivos 5
1.2 Metodologia 6
1.3 Organização do Texto 11

2 Revisão Teórica **13**
2.1 Similaridade em Cadeias e Sinais 14
2.1.1 Nomenclatura e Definições 14
2.1.2 Similaridade entre Cadeias 18
2.1.3 Similaridade entre Sinais 21
2.2 Indexação Métrica 27
2.2.1 M-Tree 29
2.2.2 Slim-Tree 38
2.3 Clusterização via Indexação métrica e k-medianas 42
2.3.1 Algoritmos que Implementam o K-medianas 44
2.3.2 Otimização do K-medianas via Estruturas de Indexação Métrica 45

3 Quantização Baseada em Extremos Relativos (QBER) e a Teoria dos Extremos Relativos **48**
3.1 Extremos Relativos 49
3.2 Extremos Relativos com Importância e Prevalência 51

3.3 Representação Baseada em Extremos Relativos 52
3.4 Montes 56
3.5 Quantização Vetorial de Montes 57
3.6 Representação Baseada em Extremos Relativos Quantizada (RBERQ) 58
3.7 Quantização Baseada em Extremos Relativos (QBER) 59

4 Implementação e Resultados **62**
4.1 Ferramentas Utilizadas 62
4.2 Componentes do Classificador 63
4.3 Fases do Modelo 64
4.3.1 Fase de pré-processamento 64
4.3.2 Fase de reconhecimento 67
4.4 Resultados 68
4.4.1 Preparação dos Dados 68
4.4.2 Preparação do Classificador 70
4.4.3 Reconhecimento dos Dados 70
4.4.4 Resultados Obtidos 71
4.4.5 Avaliação da Rentabilidade Anual 74
4.5 Avaliação da utilidade do método QBER, através da comparação de classificadores 76
4.5.1 Tamanho do sinal versus Tempo de preparação do classificador 78
4.5.2 Tamanho do sinal versus Performance do classificador . 79
4.5.3 Tamanho do sinal versus Rendimento Total 80

5 Conclusão **82**
5.1 Limitações 84
5.2 Aperfeiçoamentos Futuros 84

Referências Bibliográficas **85**

Lista de Figuras

1.1 Esquematização da implementação de referência 6
1.2 Etapas da Quantização Baseada em Extremos Relativos (QBER) 7
1.3 Fase de reconhecimento 10

2.1 Cálculo do valor da função distância unitária para as palavras *banana, badana* e *madeira, madeirite* 21
2.2 Representação de uma região do espaço métrico, a partir de um objeto de roteamento 32
2.3 Particionamento de uma M-Tree 33
2.4 Funcionamento do algoritmo Slim-Down 41

3.1 Alguns exemplos de extremos relativos 50
3.2 Máximos e mínimos relativos com prevalência 5 51
3.3 Comparação de representações em diversas prevalências 53
3.4 Seqüências de extremos relativos representando montes 56
3.5 Marcação de Montes em um sinal 57
3.6 Etapas da Quantização Baseada em Extremos Relativos (QBER) 61

4.1 Percentual de compras corretas para $confiança = 80\%$, e preparação com todos os anos anteriores 72
4.2 Percentual de compras corretas para $confiança = 60\%$, e preparação com todos os anos anteriores 72
4.3 Média de compras corretas - $confiança = 80\%$, e preparação com todos os anos anteriores 73
4.4 Média de compras corretas - $confiança = 60\%$, e preparação com todos os anos anteriores 73
4.5 Rentabilidade anual média - $confiança = 80\%$ 75
4.6 Rentabilidade anual média - $confiança = 60\%$ 75
4.7 Rentabilidade anual - melhores classificadores 76
4.8 Arquitetura de um classificador Knn Simples, sem QBER . . . 77
4.9 Tempo de Preparação versus Tamanho do Sinal 79

4.10 Percentual de Compras Corretas versus Tamanho do Sinal . . 80
4.11 Rendimento Total versus Tamanho do Sinal 81

Lista de Tabelas

2.1 Sumário para cálculo de funções, em sinais digitais 28

4.1 Código das Classes 69
4.2 Preparação dos Dados 70

Lista de Siglas

ASM Busca Aproximada de Cadeias (*Approximate String matching*)

CLARA Clustering LARge Applications

CLARANS Clustering Large Applications based on RANdomized Search

DTW Alinhamento Temporal Dinâmico (*Dynamic Time Warping*)

DSP Processamento Digital de Sinais (*Digital Signal Processing*)

EDR Distância de Edição em Seqüências Reais (*Edit Distance on Real Sequence*)

ERP Distância de Edição com Penalidade Real (*Edit Distance with Real Penalty*)

IA Inteligência Artificial

IBL Aprendizado Baseado em Instâncias (*Instance Based Learning*)

kNN k-Vizinhos mais Próximos (*k-Nearest Neighbor*)

LCSS Maior Subseqüência Comum para Sinais (*Longest Common Subsequence*)

LCS Maior Subseqüência Comum para Cadeias (*Longest Common Subsequence*)

MAM Método de Acesso Métrico

MRBER Multi-representação Baseada em Extremos Relativos

MRBERQ Multi-Representação RBER Quantizada

PAM Partitioning Around Medoids

RBER Representação Baseada em Extremos Relativos

RBERQ Representação Baseada em Extremos Relativos Quantizada

RSI Índice de Força Relativa (*Relative strength index*)

TER Teoria de Extremos Relativos

TEI Teoria dos Extremos Importantes

QBER Quantização Baseada em Extremos Relativos

Resumo

O objetivo deste trabalho é o desenvolvimento de uma nova técnica para conversão simbólica de sinais digitais, denominada Quantização Baseada em Extremos Relativos (QBER). Esta técnica pode converter sinais digitais unidimensionais em cadeias. A técnica QBER, formalizada nesta proposta, utiliza-se da Teoria de Extremos Relativos (TER) e de funções de similaridade para sinais, como a métrica Distância de Edição com Penalidade Real (*Edit Distance with Real Penalty*) (ERP). Além disso, utiliza o algoritmo de clusterização PAM-SLIM, que emprega a abordagem k-medianas, amplamente discutida na literatura. A TER é também uma contribuição deste trabalho, extendendo a Teoria dos Extremos Importantes (TEI) com o acréscimo dos conceitos prevalência, monte, Representação Baseada em Extremos Relativos (RBER) e RBERQ. Para se avaliar a utilidade da QBER desenvolveu-se um sistema de classificação de padrões de referência, baseado no classificador k-Vizinhos mais Próximos (*k-Nearest Neighbor*) (kNN). Esta implementação de referência possui as fases de pré-processamento e reconhecimento. Na fase de pré-processamento, utiliza-se a QBER para converter os objetos de treinamento do kNN em representações simbólicas. Como o classificador kNN utiliza o aprendizado baseado em instâncias (*instance-based learning)* a fase de treinamento é inexistente, sendo toda classificação baseada nos objetos de treinamento. Na fase de pré-processamento um objeto a ser avaliado também é convertido para a representação simbólica utilizada, antes de ser utilizado como entrada para o classificador kNN, na fase de reconhecimento. Com vistas a avaliar a utilidade da técnica desenvolvida são feitas comparações de seu emprego em um problema de classificação, a geração de recomendações de compra de ações. O classificador kNN implementado é então avaliado com e sem o emprego da QBER, observando-se utilidade no emprego da técnica desenvolvida, pela performance superior no tempo de preparação do classificador e na rentabilidade anual obtida.

Palavras-chave: Conversão Simbólica de Sinais Sigitais, PAM-SLIM, KNN, ERP;

Abstract

The goal is to develop a new technique for symbolic conversion of digital signals, called Quantization based on Relative Extrema (QBER). This technique can convert unidimensional digital signals into strings.

The technique QBER, formalized in this proposal, uses Relative Extrema Theory (TER) and signal's similarity functions, as metric Edit Distance with Real Penalty (ERP). In addition, also uses the clustering algorithm PAM-SLIM, which employs k-medoid approach, widely discussed in the literature. The TER is also other contribution of this work, as an extension of Important Extrema Theory, increasing concepts of prevalence, mount, Relative Extrema based Representation (RBER) and Quantized Relative Extrema based Representation (RBERQ). To evaluate the usefulness of QBER, has developed a classification system of reference patterns, based on the KNN classifier. This reference implementation has the pre-processing and recognition phases. In pre-processing, QBER is used to convert training objects of KNN in symbolic representations. As the k-Nearest Neighbor (kNN) classifier uses instance based learning, a training phase is inexistent, all classification is based on the training objects. In pre-processing phase also an object in analysis is converted to RBERQ symbolic representation, before serving as input for the classifier KNN. In order to evaluate the usefulness of the technique developed, are made comparisons in a problem of classification - generation of recommendations for the purchase of shares. The kNN classifier implemented is evaluated with and without the use of QBER, being useful in the employment by superior performance (of QBER) considering aspects of preparation time and annual return obtained.

Keywords: Symbolic Conversion of Digital Signals, PAM-SLIM, KNN, ERP;

Capítulo 1

Introdução

O cérebro é uma máquina biológica que realiza tarefas complexas. As técnicas de Inteligência Artificial (IA) tentam reproduzir nas máquinas digitais a capacidade de inferir relações, desconsiderar ruídos na representação de objetos, analisar fatos, criar regras, comparar elementos e, finalmente, elaborar teorias sobre tudo isso. Um dos objetivos da IA é entender ou construir entidades inteligentes, que poderiam imitar ou ainda superar determinadas capacidades humanas [RN95].

O foco deste trabalho é o estudo e desenvolvimento de tecnologias da IA voltadas para o reconhecimento de padrões em sinais digitais. Padrões, de maneira geral, são regularidades notáveis, que podem ser usadas para classificar objetos ou outros itens de interesse. O termo "reconhecimento de padrões"refere-se ao trabalho de agrupamento ou classificação, identificação, descrição e análise de objetos ou outras regularidades significativas através de meios automáticos ou semi-automáticos [IEE90].

Sinais são funções de uma ou mais variáveis independentes[1] que contêm informação acerca do comportamento e características de determinados fenômenos físicos. São representados matematicamente como funções de uma ou mais variáveis independentes[Lou02]. Um sinal contínuo pode ser trans-

[1]São variáveis que podem ser manipuladas. Ex: Pode ser definido o intervalo de amostragem.

formado através do processo de quantização [2] e amostragem[3] em sinais de tempo discreto ou ainda em sinais digitais.

Sinais de tempo discreto são sinais onde a variável independente (geralmente o tempo) é quantizada, isto é, tais sinais somente são definidos para valores discretos da variável independente. Matematicamente, os sinais de tempo discreto são representados como uma seqüência de números [RCH+72]. Um sinal digital implica em quantização de todas as variáveis envolvidas, isto é, quando o tempo é a variável independente e a amplitude[4] é a variável dependente[5], tem-se a amostragem da amplitude, com valores discretos, em intervalos regulares de tempo.

Os sistemas de reconhecimento de padrões, em geral, apresentam as fases de Aquisição de Dados, Pré-Processamento, Extração de Características, Escolha de Características, Escolha e Treinamento de Modelo e Avaliação. Notoriamente, a fase de Pré-Processamento inclui as tarefas de remoção de ruídos, filtragem, normalização e remoção de *outliers*[6][Pol06].

Este trabalho propõe uma nova técnica para conversão simbólica[7], denominada Quantização Baseada em Extremos Relativos (QBER) (Capítulo 3), durante a fase de Pré-Processamento, em sistemas de classificação que lidam com sinais digitais unidimensionais[8], com certa tolerância a ruídos[9].

Na área IA de, são usados diversos algoritmos para realizar a tarefa de classificar[10] ou reconhecer padrões. Entre os modelos amplamente utilizados, encontram-se as redes neurais, supervisionadas ou não, e as redes Bayesia-

[2]Quantização é o processo de representar um valor analógico (em escala contínua) em um valor digital (em escala dicreta) [App07].

[3]Amostragem é o processo de se coletar amostras em intervalos regulares de tempo [App07].

[4]Amplitude é um nome genérico que se dá à variável dependente quando não se conhece a grandeza representada[Smi02] .

[5]Ao contrário da variável independente, esta somente poder ser medida e registrada

[6]*Outliers* são elementos que não obedecem a um padrão do conjunto de dados ao qual eles pertencem [Sil04].

[7]Um processo de mudança na representação de um domínio qualquer - em nosso caso sinais digitais unidimensionais - para o domínio das cadeias.

[8]Um sinal unidimensional é um sinal de uma só variável independente. Um exemplo de sinal unidimensional é a voz, onde se tem uma função de uma variável simples, o tempo.

[9]Ruídos são componentes indesejáveis dentro de um sinal [RCH+72].

[10]É a tarefa de atribuir padrões a classes de padrões[IEE90]. Este processo muitas vezes é feito por um especialista humano. Uma boa classificação, com amostras representativas vai favorecer a identificação de objetos da população toda.

nas, sendo que estas também se valem de técnicas de Processamento Digital de Sinais (*Digital Signal Processing*) (DSP) para isolar ruídos, conforme descritas por [Pol06].

Entretanto, neste trabalho apresenta-se para avaliar a utilidade da QBER, a implementação de um sistema de classificação de padrões (Capítulo 4), baseada no classificador k-Vizinhos mais Próximos (*k-Nearest Neighbor*) (kNN), o qual utiliza uma variante do aprendizado supervisionado, o Aprendizado Baseado em Instâncias (*Instance Based Learning*) (IBL)[AKA91, Fuc97].

No IBL não existe treinamento explícito. Não se induz uma função após vários ciclos de treinamento. Simplesmente armazena-se todas os objetos de treinamento, os quais servirão de base para a tomada de decisão. O objeto a ser avaliado[11] é classificado com base nos objetos de treinamento mais similares. Assim, o IBL possui apenas duas fases distintas: pré-processamento (Seção 4.3.1) e reconhecimento (Seção 4.3.2).

O classificador kNN utiliza um conjunto de objetos de treinamento (os objetos pertencem a um determinado domínio) e uma função de similaridade entre os objetos. Cada classe de padrões é representada por um subconjunto dos objetos de treinamento[12]. Para classificar um novo objeto, o kNN simplesmente avalia a similaridade entre o objeto a ser avaliado e os objetos de treinamento, através da função de similaridade fornecida. Em seguida, são identificados os k objetos de treinamento mais próximos ao novo objeto fornecido. O novo objeto é então reconhecido como membro da classe de padrões mais freqüente nos k objetos mais similares encontrados[Pol06].

A utilização do classificador kNN pressupõe o uso de uma função de similaridade. Uma função de similaridade compara elementos de um mesmo universo ou domínio. Neste trabalho, inicialmente, o domínio é o conjunto de sinais digitais unidimensionais.

Após a aplicação da técnica QBER, concebida neste trabalho, para a fase de pré-processamento, têm-se a mudança do domínio de análise, do conjunto de sinais digitais unidimensionais para o conjunto das cadeias (Seção 2.1.1). Este processo de mudança na representação é denominado conversão

[11] É um objeto a ser classificado, em um sistema de classificação de padrões.

[12] Os objetos de treinamento são fornecidos previamente rotulados, isto é, com a identificação de suas classes. Quem faz esta rotulagem geralmente é o Supervisor do treinamento.

simbólica. As vantagens de se utilizar cadeias ao invés de sinais digitais são a possibilidade de utilizar funções de similaridade para cadeias (Seção 2.1.2), como a distância de Levenshtein, juntamente com um classificador do tipo kNN (Capítulo 4) e, a possibidade de utilização de estruturas de indexação específicas, para o domínio das cadeias.

Inicialmente, têm-se o processo de conversão simbólica através da técnica QBER e a Teoria de Extremos Relativos, ambos focos e contribuições deste trabalho. A Teoria de Extremos Relativos (TER) é uma extensão da Teoria dos Extremos Importantes (TEI)[Gan03] (Seção 3.1), que é usada para decompor os sinais analisados em componentes que passarão pelo processo de clusterização, durante o processo de conversão simbólica.

A etapa de conversão simbólica (fase de pré-processamento) desempenha um importante papel, uma vez que nesta etapa empregam-se algoritmos de IA no procedimento de quantização vetorial, como o k-médias ou k-medianas, juntamente com funções de similaridade para sinais, como a métrica Distância de Edição com Penalidade Real (*Edit Distance with Real Penalty*) [CN04]. Nessa etapa, os sinais envolvidos são transformados em cadeias ou, em outras palavras, seqüências de símbolos.

O procedimento de quantização vetorial, ou clusterização, gera centróides; tais elementos são úteis tanto na etapa de conversão simbólica como na etapa de tradução simbólica, que ocorre antes da fase de reconhecimento. A etapa de tradução simbólica realiza o pré-processamento do objeto a ser avaliado, em outras palavras, a conversão do objeto a ser avaliado, um sinal digital, em cadeias. A diferença é que não há procedimento de clusterização, porém, são utilizados os centróides obtidos na etapa de conversão simbólica.

A fase de reconhecimento realiza a busca por similaridade[13] (Seção 2.2) entre o objeto de entrada e os objetos de treinamento, representados com cadeias. Determina-se os k-vizinhos mais próximos. O objetivo desta fase é encontrar os k objetos de treinamento mais similares ao objeto a ser avaliado.

Outro aspecto interessante desta pesquisa é a otimização dos procedimentos de quantização vetorial, através do uso do algoritmo PAM-SLIM [BRTJ06] e, ainda, a utilização da estrutura de indexação métrica Slim-Tree [TTSF00] para otimizar o reconhecimento do classificador kNN, o qual tra-

[13]É a execução de consulta por similaridade, como a consulta por abrangência e a consulta k-vizinhos mais próximos (Seção 2.2).

balha com cadeias e com a distância de Levenshtein (Seção 2.1.2).

1.1 Justificativa e Objetivos

A técnica QBER, proposta neste trabalho, justifica-se por ser uma forma alternativa de realizar conversão simbólica de sinais digitais, isto é, transformá-los em cadeias. A utilização desta técnica permite a utilização de modelos computacionais que lidam somente com cadeias.

Para definir a técnica QBER, ocorreram diversas inovações, entre elas a definição do conceito de *Monte* (Seção 3.4), o qual permitiu a integração de várias técnicas para viabilizar a comparação de partes dos sinais analisados, via clusterização através do algoritmo PAM-SLIM [BRTJ06] (Seção 2.3.2). Outra inovação relaciona-se à definição do parâmetro *prevalência*, o que permite aumentar ou diminuir o nível de sensibilidade, para a identificação dos sub-componentes dos sinais, os *Montes*.

Os conceitos de monte e prevalência aperfeiçoaram a TEI, no sentido que permitiram a definição da Representação Baseada em Extremos Relativos (RBER) (Seção 3.3) e da Representação Baseada em Extremos Relativos Quantizada (RBERQ) (Seção 3.6), resultando por fim no processo QBER (Seção 3.7), para transformação de sinais digitais em cadeias.

Finalmente, para comprovar a utilidade da técnica QBER, desenvolveu-se uma implementação de referência, um classificador de padrões, no estilo kNN, que usa a técnica idealizada, em sua fase de pré-processamento. Assim, tem-se o emprego do classificador kNN (Capítulo 4), para classificação de padrões através da execução da consulta aos k-vizinhos mais próximos (Seção 2.2), usando a distância de Levenshtein (Seção 2.1.2), tendo como objetos de treinamento um conjunto de cadeias e como objeto a ser avaliado, uma cadeia também.

A análise do classificador implementado é realizada através da definição de um exemplo de aplicação real - recomendações de compra de ações - com validação de sua classificação, através de simulação do uso, em várias datas, considerando os dados anteriores à data.

1.2 Metodologia

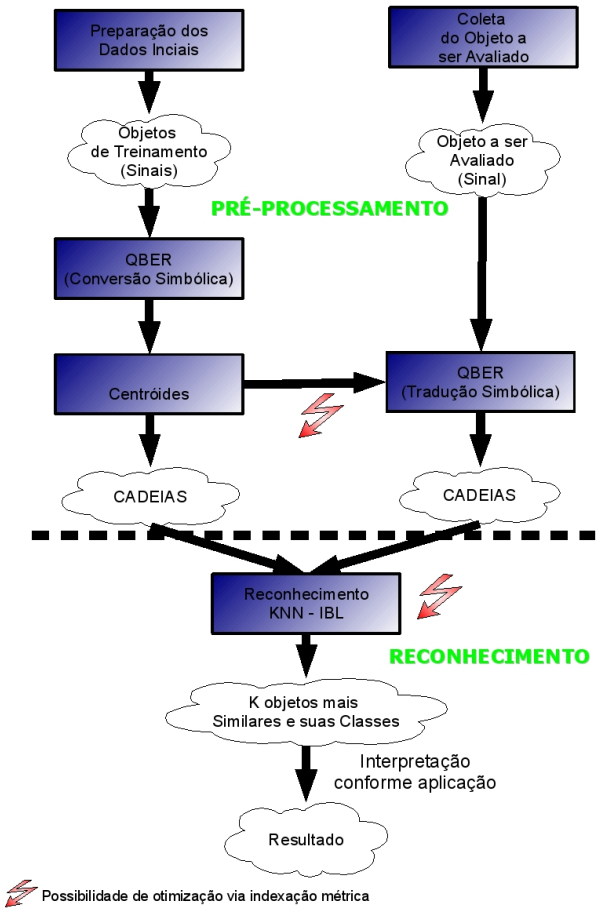


Figura 1.1: Esquematização da implementação de referência

Nesta seção serão descritos todos os passos realizados para que o objetivo deste trabalho fosse atingido. Na Figura 1.1 se observam as duas fases da implementação de referência: Fase de pré-processamento e fase de reconhecimento.

Os processos de Conversão Simbólica e Tradução Simbólica encontram-se melhor detalhados na Figura 1.2. Estas etapas utilizam a técnica QBER (Seção 3.7) para converter os sinais digitais em cadeias. Os dois processos são muito parecidos; a diferença está na sub-etapa de Clusterização, presente na conversão e ausente na tradução.

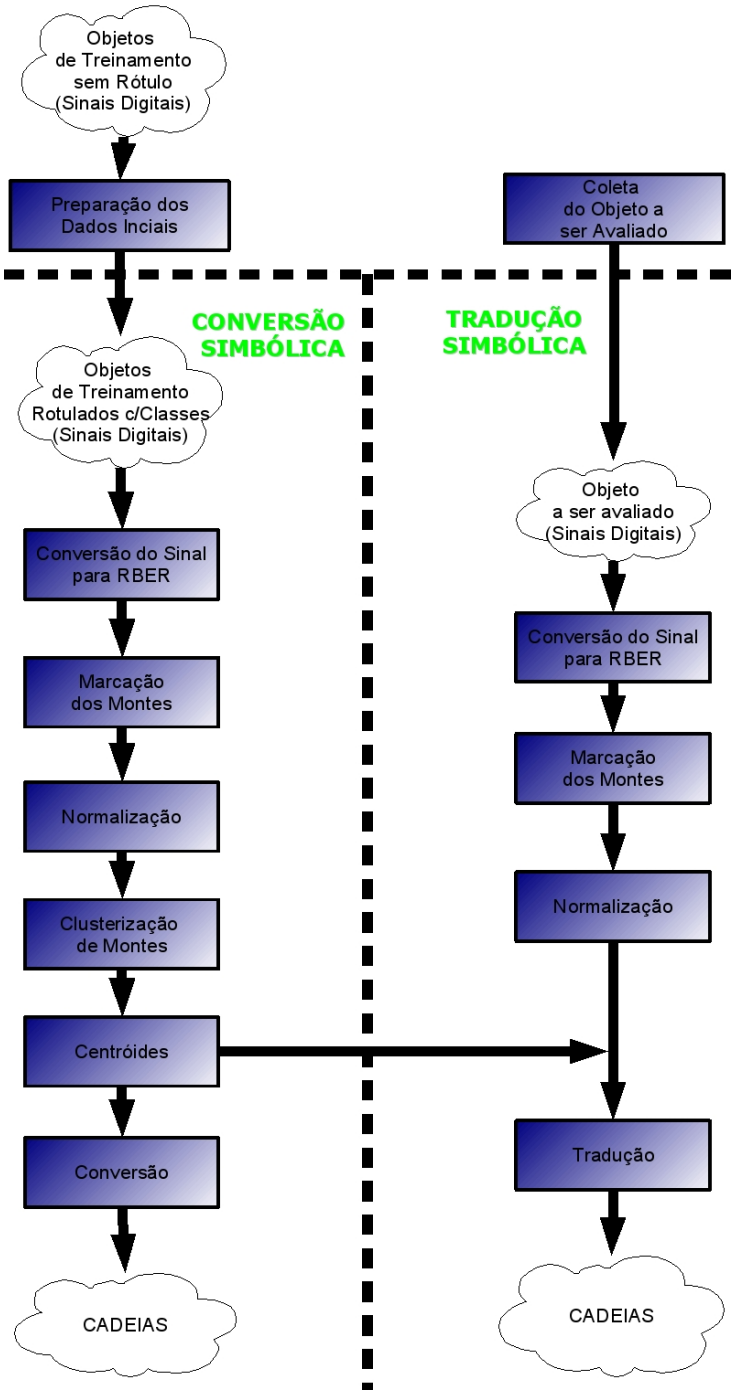


Figura 1.2: Etapas da Quantização Baseada em Extremos Relativos (QBER)

Inicialmente, os objetos de treinamento são coletados e rotulados em uma classe específica. Este processo pode variar conforme a aplicação, portanto o classificador em si não precisa efetuar esta designação(rótulo de classe). O que ocorre é a importação dos objetos de treinamento via um arquivo XML padronizado[14], assim, o mesmo classificador pode ser usado para diferentes aplicações. Um programa externo pode gerar, se necessário, o arquivo XML para importação pelo classificador. Além disso, um indicador[15] pode ser utilizado (Seção 4.3.1).

Os objetos de treinamento, após a importação, são armazenados em um banco de dados do classificador, para posterior utilização. No exemplo apresentado neste trabalho - recomendações de compra de ações - um módulo específico da aplicação gera o arquivo XML dos objetos de treinamento. Outro detalhe importante é que podem ser escolhidos subconjuntos dos sinais cadastrados no banco de dados, como objetos de treinamento.

Outro processo é a Coleta do Objeto a ser Avaliado. Pode ser simplesmente fornecido um arquivo contendo o objeto a ser classificado, em determinado formato, ou pode ser solicitado do classificador o reconhecimento de um objeto já cadastrado no banco de dados. Isto é útil para o processo de validação do classificador. É claro que em nosso exemplo de aplicação, como recomenda a literatura, o objeto a ser classificado não está entre os objetos de treinamento.

A Conversão Simbólica se sub-divide nas seguintes etapas:

1. Conversão do Sinal para RBER: Os sinais envolvidos têm seus extremos relativos (Seção 3.1) identificados. O parâmetro desta etapa é o valor da prevalência (Definição 11);

2. Marcação de Montes: Nesta fase são identificados sub-componentes dos sinais envolvidos, os montes (Seção 3.4);

3. Normalização: Cada *Monte* gerado, para cada objeto de referência é normalizado. O procedimento de normalização varia conforme o problema atacado;

[14]O XML é um formato padrão para representação de informações, facilmente codificado e utilizado em inúmeros sistemas e aplicações.

[15]Uma transformação matemática aplicada em sinais. No jargão da análise técnica de ações, é a aplicação de uma filtro ou transformação nas cotações de uma determinada ação.

4. Quantização de Montes (Clusterização) : Nesta etapa os *Montes* da etapa anterior são quantizados. Para se realizar esta quantização pode ser utilizado o algoritmo PAM-SLIM, com a função de distância ERP (Seções 2.1.3 e 2.3.2). Os centróides, produtos da clusterização, são armazenados;

5. Conversão: Cada monte, em cada sinal, é comparado com uma lista de centróides gerada no passo anterior. Associando-se cada centróide a um símbolo, cada sinal pode ter seus montes traduzidos para o símbolo associado à centróide mais similar. Assim, cada sinal é traduzido para uma cadeia, composta do símbolo associado ao centróide.

A Tradução Simbólica é a mesma coisa que a Conversão Simbólica. A diferença é que não existe a sub-etapa Quantização de Montes. Também a etapa *Conversão* é denominada *Tradução* a qual utiliza as centróides previamente armazenadas na etapa de Conversão Simbólica.

A Figura 1.3 apresenta com mais detalhes a fase de reconhecimento, visualizando-se os processos de preparação e otimização do classificador kNN, a saída da classificação e a interpretação desta saída para geração do resultado.

A fase de Reconhecimento tem como entrada os objetos de treinamento - convertidos em cadeias - e o objeto a ser avaliado - também convertido em cadeia. Inicialmente tem-se o processo de preparação e otimização do classificador kNN; tal etapa poderia ser desprezada, no entanto é muito valiosa em virtude de melhorar muito a performance do classificador. Nesta etapa, é gerada a árvore de indexação métrica (Seção 2.2.2) SlimTree dos objetos de treinamento, utilizando a distância de Levenshtein, uma métrica para cadeias (Seção 2.1.2). Isto otimiza a busca dos objetos de treinamento mais similares ao objeto a ser avaliado. O produto desta etapa é o classificador kNN otimizado.

A partir do instante que o classificador kNN está devidamente preparado e otimizado, podem ser fornecidos objetos para classificação. Um objeto a ser avaliado, fornecido ao classificador kNN tem como resposta os k objetos de treinamento mais similares (pela distância de Levenshtein), com seus respectivos rótulos de classe.

O último passo interpreta a saída do classificador para gerar um resultado que possa ser utilizado. Em geral, o objeto a ser avaliado é reconhecido como

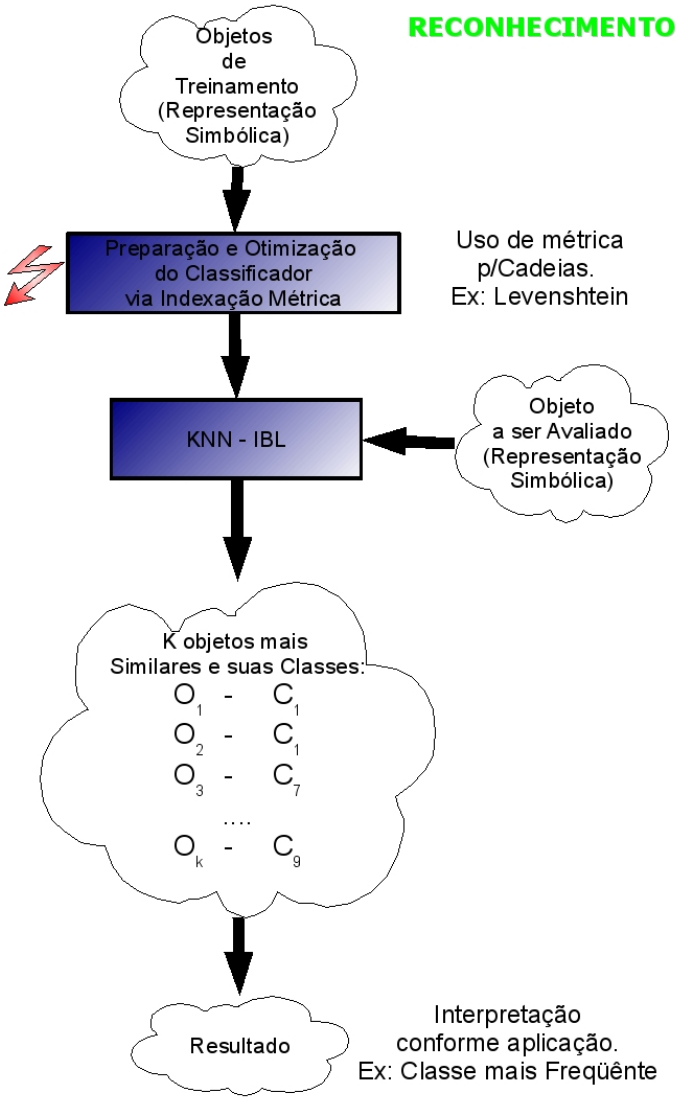


Figura 1.3: Fase de reconhecimento

pertencente à classe mais freqüente nos k objetos de treinamento mais similares. Porém, em nosso exemplo real, a interpretação da saída do classificador é outra. O alvo é gerar recomendações de compra de ações que tenham lucro $\geq 2\%$ nos 3 dias que sucedem ao dia analisado.

Um dia a ser analisado é composto de suas cotações de abertura, fechamento, máxima e mínima e de n dias anteriores. Em nosso caso, fizemos o teste com 14 e 20 dias anteriores respectivamente. Os objetos de treinamento têm o mesmo formato, porém já se sabe de antemão qual foi a rentabilidade máxima, nos 3 dias posteriores. As classes dos objetos de treinamento são $L_$,$L0$,$L1$,$L2$,$L3$,$L4$,$L5$,$L6$,$L8$,$L9$,$L10$, sendo usadas para lucros inexistentes, lucro máximo de 1% nos três dias posteriores e assim sucessivamente até lucro máximo acima de 10%, em três dias posteriores. Estes cálculos serão explicitados na Seção 4.4.

A saída do classificador são os códigos dos objetos de treinamento, com o respectivo rótulo de classe. Tem-se um parâmetro denominado confiança, que serve para interpretar a saída do classificador. Se a confiança é 80%, por exemplo, tem-se que é recomendado compra se 80% dos k objetos de treinamento mais similares tiveram lucro máximo $\geq 2\%$, em outras palavras, 80% dos k objetos de treinamento mais similares têm como rótulo uma das classes $L2$,$L3$,$L4$,$L5$,$L6$,$L8$,$L9$,$L10$. Analogamente à preparação dos objetos de treinamento, a interpretação dos dados é feita por módulos especializados e poderia ser feita externamente, por outra aplicação.

1.3 Organização do Texto

Este Capítulo apresentou uma introdução ao tema da dissertação, juntamente com os objetivos e a metodologia para atingi-los.

O Capítulo 2 apresenta uma Revisão Teórica dos conceitos e métodos utilizados na concepção da TER e QBER. A Seção 2.1 apresenta o conceito de similaridade, com a definição de funções de similaridade, tanto no domínio dos sinais, quanto no domínio das cadeias. A utilização de funções de similaridade é fundamental para etapa de Clusterização, na fase de pré-processamento e ainda para a fase de reconhecimento, através do classificador kNN. A Seção 2.2 expõe a técnica de Indexação Métrica, muito utilizada na implementação e otimização do método proposto. A otimização é realizada

na etapa de Clusterização e na fase de reconhecimento, nesta última, otimizando a busca de objetos de treinamento mais similares. Já a Seção 2.3 apresenta o processo de clusterização via indexação métrica e k-medianas, utilizados na etapa de Clusterização de Montes, nos procedimentos de Conversão Simbólica e Tradução Simbólica.

O Capítulo 3 apresenta a Teoria de Extremos Relativos, juntamente com a QBER, ambos foco e contribuição deste trabalho.

O Capítulo 4 expõe o desenvolvimento da implementação de referência, ferramenta utilizada na avaliação do método QBER, como representação útil. Além disso, expõe os resultados obtidos na avaliação da implementação de referência, através de uma aplicação exemplo, um sistema de recomendação de compra de ações. Também neste capítulo é feita a comparação do classificador desenvolvido, com outro classificador que não utiliza o método QBER.

Finalmente, o Capítulo 5 apresenta as conclusões deste projeto, os objetivos alcançados, bem como as sugestões de futuras linhas de pesquisa.

Capítulo 2

Revisão Teórica

Este trabalho se apoia em diversas teorias e conceitos já definidos, testados e implementados em outras pesquisas. São conceitos provenientes de áreas como a Biologia Computacional, a Teoria da Computação, o Processamento Digital de Sinais e, finalmente a Inteligência Artificial.

Os conceitos de similaridade em cadeias são muitos utilizados e pesquisados na área de Biologia Computacional e Teoria da Computação. Os conceitos de similaridade em sinais são provenientes da área de Processamento Digital de Sinais. Já os conceitos de clusterização e o classificador kNN são foco de muitas pesquisas na área de Inteligência Artificial. Por último, as estruturas e algoritmos de indexação métrica são provenientes da sub-área de Banco de Dados - Indexação e Recuperação de Dados Complexos.

Dessa forma, nas seções subseqüentes, serão expostos os conceitos de similaridade em sinais e em cadeias, a técnica e estruturas de Indexação Métrica, a Clusterização do tipo k-medianas, através do algoritmo PAM-SLIM, e, finalmente o classificador de padrões kNN, todos fundamentais para a aplicação da técnica QBER, bem como para o desenvolvimento da implementação de referência.

2.1 Similaridade em Cadeias e Sinais

O conceito de similaridade é amplo. Depende essencialmente da função de similaridade, ou função de distância, utilizada. Relacionado ao conceito de função de distância estão os conceitos de métrica, pseudo-métrica, semi-métrica e quasi-métrica, detalhados na seção 2.1.1. Toda métrica é uma função de distância mas, nem toda função de distância é uma métrica. O mesmo vale para os conceitos de pseudo-métrica, semi-métrica e quasi-métrica. Existem diversos tipos de funções de distância; no campo dos sinais temos as funções Alinhamento Temporal Dinâmico (*Dynamic Time Warping*) (DTW), Distância de Edição com Penalidade Real (*Edit Distance with Real Penalty*) (ERP), Maior Subseqüência Comum para Sinais (*Longest Common Subsequence*) (LCSS), Distância de Edição em Seqüências Reais (*Edit Distance on Real Sequence*) (EDR) [MP07], apresentadas na seção 2.1.3; já no domínio das cadeias, existem a distância Levenshtein ou unitária, a distância de Hamming e a distância LCS, abordadas na seção 2.1.2.

O conceito de similaridade também depende do tipo de informação a ser analisado. Dessa forma, faz-se necessário definir a noção de similaridade entre sinais digitais unidimensionais, e ainda a similaridade entre cadeias de caracteres.

2.1.1 Nomenclatura e Definições

A noção de similaridade é oposta ao conceito de dissimilaridade; quanto maior a similaridade menor a dissimilaridade. Uma *função de distância*, ou distância, é uma medida de dissimilaridade entre dois elementos de um determinado conjunto, ou domínio D. Assim, uma função de distância d para o domínio D é do tipo $d : D \times D \to \mathbb{R}$. Em termos formais uma função de distância pode ser expressa conforme a definição 1 [vL04]:

Definição 1 *Seja D um conjunto e $d : D \times D \to \mathbb{R}$, uma função de distância. A função d satisfaz as seguintes propriedades:*

$$d(x, x) = 0 \tag{1a}$$
$$d(x, y) \geq 0 \text{ } \textit{(não negatividade)} \tag{1b}$$

Uma função de distância pode se encaixar no formalismo matemático de métrica, podendo ser denominada métrica. A Definição 2 enumera as propriedades necessárias a uma métrica [Ska08].

Definição 2 *Seja D um conjunto e $d : D \times D \rightarrow \mathbb{R}$, uma função de distância. Dados $x, y, z \in D$, a função d é denominada métrica se satisfaz as seguintes propriedades:*

$$d(x,y) \geq 0 \ \textit{(não negatividade)} \tag{2a}$$
$$d(x,x) = 0 \ \textit{(reflexividade)} \tag{2b}$$
$$d(x,y) \neq 0, \ \textit{se } x \neq y \tag{2c}$$
$$d(x,y) = d(y,x) \ \textit{(simetria)} \tag{2d}$$
$$d(x,z) \leq d(x,y) + d(y,z) \ \textit{(desigualdade triangular)} \ . \tag{2e}$$

Se uma função de distância satisfaz todas propriedades da definição 2, ela é denominada métrica; além disso, o par ordenado $< D, d >$ é um espaço métrico. Se todas as propriedades são válidas exceto a propriedade 2c, então tem-se uma pseudo-métrica; Caso somente a propriedade 2d seja inválida, tem-se uma quasi-métrica; Se somente a propriedade 2e é inválida, tem-se uma semi-métrica[Pek05, vL04, Sko07, Ska08].

Mais à frente, nesta Seção, serão estudadas funções de distância, métricas ou não, para o domínio das cadeias e para o domínio dos sinais digitais unidimensionais.

Cadeias

O conceito de cadeia é relacionado aos conceitos de símbolo e alfabeto. Um *símbolo* é uma entidade abstrata tal como o conceito de "ponto"e "linha"não definidos em geometria. Letras e dígitos são exemplos de símbolos freqüentemente usados[HU90]. Um *alfabeto* é um conjunto de símbolos. Neste trabalho denotamos um alfabeto por $\Sigma = \{a_1, a_2, ..., a_n\}$, onde a_i é um símbolo para $i = 1, 2, ..., n$.

Uma *palavra*, ou cadeia, é formada usando os símbolos de Σ. Mais precisamente, uma palavra é uma seqüência finita de símbolos justapostos. Um exemplo de palavra é *"banana"* composta dos símbolos *b,a,n*. Uma palavra w tem tamanho $|w|$, isto é, w é a justaposição de $|w|$ símbolos; em nosso exemplo $|banana| = 6$. Um caso particular de palavra, representada pelo símbolo ϵ, é a *palavra vazia*, que não contém nenhum símbolo, assim $|\epsilon| = 0$.

O conjunto de todas as palavras possíveis em um alfabeto Σ é dado por :

$$\Sigma^* = \{a_1 a_2 ... a_m | a_k \in \Sigma, k \leq m, m \in \mathbb{N}, m \neq 0\} \cup \{\epsilon\} \tag{3}$$

Os subconjuntos de Σ^* que contêm palavras de mesmo comprimento m são denotados por Σ^m, $\forall m \in \mathbb{Z}^+$. Em particular, quando $m = 0$, temos $\Sigma^0 = \{\epsilon\}$, representando todas as palavras de tamanho zero. Neste texto, $a_1, a_2, ..., a_n$ representam símbolos; já w, u, v representam palavras.

Definição 3 *(Concatenação) Sejam as cadeias* $u = a_1 a_2 ... a_{|u|}$ *e* $w = b_1 b_2 ... b_{|w|}$. *A operação de concatenação u.w ou simplesmente uw, resulta na cadeia* $uw = a_1 a_2 ... a_{|u|} b_1 b_2 b_{|w|}$. *A concatenação de duas cadeias u e w é a cadeia formada primeiramente pelos símbolos de u, seguidos pelos símbolos de w. A justaposição é o operador da operação de concatenação. Assim, uw é a concatenação das cadeias u e w. Esta operação também pode ser escrita como u.w . Além disso,* $|u.w| = |u| + |w|$.

Como exemplo para concatenação se $u = banana$ e $w = madura$, então $uw = bananamadura$.

Uma *subseqüência* de u é uma palavra que pode ser obtida através da remoção de símbolos de u. Já um *fator de u* é uma subseqüência de u obtida através da remoção de símbolos sucessivos no final ou no começo da palavra u. Por exemplo, dado $u = \{a_1, a_2, ..., a_{|u|}\}$, e w fator de u, $w = \{a_i, a_{i+1}, ..., a_j\}$ para $1 \leq i \leq j \leq |u|$. Outra forma de representar w é $u_{i,j}$, $1 \leq i \leq j \leq |u|$.

Um *prefixo w* de uma palavra u é uma subseqüência de u obtida através da remoção de símbolos sucessivos do final de u, isto é, se $u = \{a_1, a_2, ..., a_{|u|}\}$, $w = \{a_1, a_2, ..., a_{|w|}\}$, com $|w| \leq |u|$. Já um *sufixo w*, de uma palavra u é tal que se $u = \{a_1, a_2, ..., a_{|u|}\}$, $w = \{a_j, a_{j+1}, ..., a_{|u|}\}$, com $j \geq 1$. Em outras

palavras, um prefixo w de u é um fator $u_{1,j}$, com $j \leq |u|$; já um sufixo w de u é um fator $u_{j,|u|}$, com $1 \leq j \leq |u|$.

Uma função de distância entre cadeias é uma função $d : \Sigma^* \times \Sigma^* \rightarrow \mathbb{R}$ para a qual podem valer as propriedades da definição 2, podendo ela ser denominada métrica[Nav01], semi-métrica ou pseudo-métrica, entre cadeias, conforme o caso.

Sinais Digitais Unidimensionais

Sinais são funções de uma ou mais variáveis independentes que contêm informação acerca do comportamento e características de determinados fenômenos físicos. São representados matematicamente como funções de uma ou mais variáveis independentes[Lou02].

Este texto trabalha com sinais digitais unidimensionais, que são sinais em que todas variáveis envolvidas são quantizadas e, além disso só possuem uma variável independente. Assim, neste texto, os sinais digitais unidimensionais são representados por letras maiúsculas tais como X,Y,Z que contém seqüências de valores reais. É importante ressaltar que os valores, como são quantizados, possuem uma representação discreta, contável. Assim, um sinal X com n amostras igualmente espaçadas no tempo(n medições) é representado como $X = \{x_1, x_2, ..., x_n\}$; onde $x_i \in \mathbb{R}$, $i \leq n$, representa um valor real, na amostragem i. Também, neste texto, $|X| = n$.

É importante ressaltar que como os sinais estudados neste texto são sinais digitais unidimensionais, podemos, sem perda de generalidade, utilizar o nome *série temporal*. Uma série temporal S, segundo [CN04], é definida como uma seqüência de valores reais. Dessa forma, $S = [s_1, s_2, ..., s_n]$, com s_i real para $1 \leq i \leq n$, é uma representação da série temporal S. Mais especificamente, a sua representação natural.

Uma função de distância entre sinais digitais unidimensionais é uma função $d : \mathbb{S} \times \mathbb{S} \rightarrow$ R, onde $\mathbb{S}$ é o domínio dos sinais digitais unidimensionais, e para a qual podem valer as propriedades da definição 2, podendo a mesma ser denominada métrica, semi-métrica, pseudo-métrica entre sinais digitais unidimensionais, conforme o caso.

Neste trabalho, os conceitos de cadeias e sinais são relacionados, uma vez

que, através da técnica QBER, sinais digitais unidimensionais são transformados em cadeias. A vantagem de utilizar cadeias, ao invés de sinais digitais, para manipulação, é o emprego de funções de distância para cadeias, amplamente estudadas na literatura, na classificação de padrões.

2.1.2 Similaridade entre Cadeias

Existem diversas funções de distância para medir a dissimilaridade entre cadeias. Um modelo amplamente empregado para definição de funções de similaridade para cadeias é o modelo da distância de edição. Tal modelo emprega a noção de quantidade mínima de operações para transformar uma cadeia em outra. As operações possíveis são, em geral, a inserção, a remoção e a alteração. A cada operação é atribuído um peso. Assim, uma função de distância baseada no modelo de distância de edição define pesos específicos para cada operação; para seu cálculo basta achar o valor mínimo que pode ser atingido em sequências de operações que transformar uma cadeia em outra.Uma forma de fazer isto é utilizar técnicas de programação dinâmica.

Operações, alinhamentos e função δ

Uma *operação de edição* é um par ordenado $(\alpha, \beta) \in (\Sigma^1 \cup \{\epsilon\}) \times (\Sigma^1 \cup \{\epsilon\}) \backslash \{(\epsilon, \epsilon)\}$. A operação (α, β) é usualmente escrita como $\alpha \rightarrow \beta$. Isto reflete a visão operacional que considera as operações de edição como regras para transformação de uma cadeia u em uma cadeia v. Existem vários tipos de operação de edição [Kur01]:

- Inserção: $\epsilon \rightarrow a$, isto é, inserção da letra a.
- Remoção: $a \rightarrow \epsilon$, isto é, remoção da letra a.
- Substituição: $a \rightarrow b$, isto é, substituição da letra a por b.

Um *alinhamento* A de u e v é uma seqüência

$$(\alpha_1 \rightarrow \beta_1, ..., \alpha_h \rightarrow \beta_h)$$

de operações de edição tal que $u = \alpha_1, ..., \alpha_h$ e $v = \beta_1, ..., \beta_h$.

É interessante ressaltar que são válidas operações de remoção e inserção. Assim, por exemplo, considerando-se as cadeias $u = bcacd$ e $v = dbadad$, um alinhamento A de u em v poderia ser $A = (\epsilon \rightarrow d, b \rightarrow b, c \rightarrow a, \epsilon \rightarrow d, a \rightarrow a, c \rightarrow \epsilon, d \rightarrow d)$. Existem alinhamentos ótimos, isto é, existem alinhamentos que representam menor custo. A noção de alinhamento ótimo depende, portanto, da noção de custo.

Uma *função de custo* δ atribui a cada operação de edição $\alpha \rightarrow \beta$, um custo real positivo. Tem-se que $\delta(\alpha \rightarrow \alpha) = 0$. Se $\delta(\alpha \rightarrow \beta) = \delta(\beta \rightarrow \alpha)$ para todas as operações de edição $\alpha \rightarrow \beta$ e $\beta \rightarrow \alpha$, então δ é simétrico. Se $\delta(\alpha \rightarrow \beta) = 1$, para todas operações de edição, com $\alpha \neq \beta$, então δ é uma função de custo unitária.

A função δ é extendida para alinhamentos de uma maneira direta: O custo de um alinhamento $A = (\alpha_1 \rightarrow \beta_1, ..., \alpha_h \rightarrow \beta_h)$ é a soma dos custos de todas as operações de edição. Em outras palavras:

$$\delta(A) = \sum_{i=1}^{h} \delta(\alpha_i \rightarrow \beta_i).$$

Funções de distância unitária, de Hamming e LCS

Agora já podem ser definidas as funções de custo utilizadas em diferentes funções de distância para cadeias. Tais funções utilizam o modelo da distância de edição, isto é, uma função de custo, operações de edição e alinhamentos.

A *distância de edição* entre u e v, denotada por $edist_\delta(u, v)$, é o custo mínimo possível de um alinhamento entre u e v. Isto é,

$$edist_\delta(u, v) = min\{\delta(A) | A \text{ é um alinhamento de } u \text{ e} v\}.$$

Um alinhamento A de u e v é ótimo se $\delta(A) = edist_\delta(u, v)$. O problema da distância de edição é calcular a distância entre u e v.

A *distância de Levenshtein*, ou distância de edição unitária, tem sua

função de custo definida pela função δ de custo unitário:

$$\delta(\alpha \rightarrow \beta) = \begin{cases} 0 & \text{Se } \alpha, \beta \in \Sigma \text{ e } \alpha = \beta \\ 1 & \text{caso contrário} \end{cases} \tag{4}$$

A *distância de hamming* tem sua função de custo definida por $\delta_{hamming}$, a qual não considera inserções e remoções:

$$\delta_{hamming}(\alpha \rightarrow \beta) = \begin{cases} 0 & \text{Se } \alpha, \beta \in \Sigma \text{ e } \alpha = \beta \\ 1 & \text{Senão, se } \alpha, \beta \in \Sigma \text{ e } \alpha \neq \beta \\ \infty & \text{caso contrário}(\alpha = \epsilon \text{ ou } \beta = \epsilon) \end{cases} \tag{5}$$

A distância LCS tem sua função de custo definida por δ_{LCS}:

$$\delta_{LCS}(\alpha \rightarrow \beta) = \begin{cases} 0 & \text{Se } \alpha, \beta \in \Sigma \text{ e } \alpha = \beta \\ 2 & \text{Senão, se } \alpha, \beta \in \Sigma \text{ e } \alpha \neq \beta \\ 1 & \text{caso contrário}(\alpha = \epsilon \text{ ou } \beta = \epsilon) \end{cases} \tag{6}$$

d

Cálculo do valor da função de distância unitária, usando programação dinâmica

Um algoritmo para resolver o problema, baseando-se na programação dinâmica pode ser visto em [Sel80]. Apesar de existirem alternativas mais eficientes, ele tem sua utilidade na possibilidade de utilização de diferentes funções de distância.

Para calcular a distância de edição unitária, por exemplo, $edist_\delta(u, v)$, basta preencher uma matriz $C_{0..|u|,0...|v|}$. A posição $C_{|u|,|v|}$ é preenchida como o número mínimo de operações necessários para transformar $u_1u_2..u_{|u|}$ em $v_1v_2..v_{|v|}$. Assim tem-se que:

$$C_{i,0} = i \tag{7}$$

		B	A	N	A	N	A
	0	1	2	3	4	5	6
B	1	0	1	2	3	4	5
A	2	1	0	1	2	3	4
D	3	2	1	1	2	3	4
A	4	3	2	2	1	2	3
N	5	4	3	2	2	1	2
A	6	5	4	3	2	2	1

		M	A	D	E	I	R	A
	0	1	2	3	4	5	6	7
M	1	0	1	2	3	4	5	6
A	2	1	0	1	2	3	4	5
D	3	2	1	0	1	2	3	4
E	4	3	2	1	0	1	2	3
I	5	4	3	2	1	0	1	2
R	6	5	4	3	2	1	0	1
I	7	6	5	4	3	2	1	1
T	8	7	6	5	4	3	2	2
E	9	8	7	6	5	4	3	3

Figura 2.1: Cálculo do valor da função distância unitária para as palavras *banana, badana* e *madeira, madeirite*

$$C_{0,j} = j \tag{8}$$

$$C_{i,j} = \begin{cases} C_{i-1,j-1} & \text{se } u_i = v_j \\ 1 + Min(C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1}) \end{cases} \tag{9}$$

A Figura 2.1 demonstra o preenchimento da matriz C. A execução do algoritmo em questão é da complexidade de $O(|u||v|)$ no pior caso. O espaço utilizado é $O(min(|u|, |v|))$ no pior caso, pois só a coluna anterior precisa ser armazenada.

2.1.3 Similaridade entre Sinais

Analogamente à similaridade de cadeias, a similaridade de sinais está estritamente relacionada às funções de distância para sinais. Existem basicamente duas classes de funções de distância. A primeira classe inclui funções baseadas nas normas L1 e L2 como as funções de distância DTW e ERP. A segunda classe é composta de funções de distância que calculam um índice de similaridade baseando-se em um limiar de encaixe ε [MP07]; exemplos desta segunda classe de funções de distância são LCSS e EDR.

O cálculo das funções de distância apresentadas podem ser realizados através de técnicas de programação dinâmica, conforme demonstrado na seção anterior.

Funções de distância L_1 e L_2, matriz de custo, função *Rest* e caminho de alinhamento

Para efeito do cálculo de funções de distância para sinais, existem alguns conceitos básicos, utilizados amplamente. Estes conceitos são as funções de distância L_1 e L_2, a matriz de custo, a função Rest e o conceito de caminho de alinhamento.

A *distância* L_1, ou *distância de Manhanttan*, entre duas séries temporais $X = \{x_1, x_2, ..., x_n\}$ e $Y = \{y_1, y_2, ..., y_n\}$ é expressa por:

$$Dist_{L1}(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{10}$$

A *distância* L_2, ou *distância euclidiana*, entre duas séries temporais $X = \{x_1, x_2, ..., x_n\}$ e $Y = \{y_1, y_2, ..., y_n\}$ é expressa por:

$$Dist_{L2}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{11}$$

A *função Rest* de uma série temporal X é definida por:

$$Rest(X) = x_2, ..., x_{|X|} \tag{12}$$

Já a *matriz de custo* D, entre duas séries temporais X e Y, é uma matriz de tamanho $|X| \times |Y|$ tal que $D(i, j) = Dist(x_i, y_j)$, onde Dist é qualquer função de distância que se tenha interesse utilizar, definida no domínio das séries temporais.

O conceito de *caminho de alinhamento* W é definido considerando-se uma matriz de custo M, para duas séries temporais X e Y. Assim, $W = w_1, w_2, ..., w_{|W|}$ (*warping path*) é tal que $w_k = (i, j)$ para $1 \leq i \leq |X|$ e $1 \leq j \leq |Y|$. Além disso $max(|X|, |Y|) \leq |W| \leq |X| + |Y|$, onde $|W|$ é o tamanho do caminho de alinhamento. Também dados $w_k = (i, j)$ e $w_{k+1} = (i', j')$ então, $i \leq i' \leq i + 1$ e $j \leq j' \leq j + 1$.

Os conceitos de matriz de custo e caminho de alinhamento são relacionados ao conceito de alinhamento do domínio das cadeias e a forma de cálculo vistos na seção 2.1.2.

Alinhamento Temporal Dinâmico (*Dynamic Time Warping*)

O Alinhamento Temporal Dinâmico - DTW - é uma técnica que encontra um alinhamento ótimo entre duas séries temporais, esticando-se ou encolhendo-se uma das séries. Este alinhamento entre duas séries temporais pode ser usado para encontrar regiões correspondentes entre duas séries temporais ou ainda para determinar a similaridade entre elas. [SP04].

Para se calcular a DTW é necessário o cálculo da matriz de custo. Após se realizar o cálculo da matriz de custo, deve-se achar o menor caminho de alinhamento. Dadas duas séries temporais X e Y, e uma matriz de custo M, entenda-se por menor caminho de alinhamento o caminho de menor custo, ou caminho de alinhamento ótimo. Portanto deve-se achar um caminho do ponto $M(1,1)$ até o ponto $M(|X|,|Y|)$ que tenha custo mínimo. O índice de similaridade seria então o custo total do caminho de alinhamento. Assim, o cálculo da DTW é expresso por:

$$DTW(X,Y) = \begin{cases} 0\ , & \text{se } |X| = |Y| = 0 \\ \infty\ , & \text{se } |X| = 0 \text{ ou } |Y| = 0 \\ Dist(x_1, y_1) + & min \begin{cases} DTW(Rest(X), Rest(Y)) \\ DTW(Rest(X), Y) \\ DTW(X, Rest(Y)) \end{cases} \text{, caso contrário} \end{cases} \tag{13}$$

Distância de Edição em Seqüências Reais (*Edit Distance on Real Sequence*)

A função de distância EDR[COO05] se baseia no modelo de distância de edição do domínio das cadeias. A função de distância entre dois símbolos r_i,s_i das cadeias R e S, respectivamente, pode ser representada por:

$$dist(r_i, s_i) = \begin{cases} 0 & \text{Se } r_i = s_i \\ 1 & \text{Caso contrário} \end{cases} \tag{14}$$

Na EDR, a equação 14 é alterada para se adequar ao conceito de séries temporais. Assim, no caso de cadeias, r_i e s_i são símbolos; no domínio de séries temporais r_i e s_i são números reais ou ainda vetores n-dimensionais. A equação 14 é reescrita para o domínio de séries temporais, onde se faz um relaxamento do conceito de igualdade, introduzindo um limiar ε. Esta função, $Dist_{edr}$, considera os elementos x_i e y_i como i-ésimos elementos das séries temporais X e Y, respectivamente, e é expressa por:

$$Dist_{edr}(x_i, y_i) = \begin{cases} 0\ , & \text{Se } |x_i - y_i| \leq \varepsilon \\ 1\ , & \text{Caso contrário} \end{cases} \tag{15}$$

A partir da definição da função de distância para séries, na equação 15, uma generalização da distância Levenshtein - Equação 4, para séries temporais, é possível definir com clareza a função de distância EDR.

A função de distância $EDR(X, Y)$, para duas séries temporais $X = \{x_1, x_2, ..., x_{|X|}\}$ e $Y = \{y_1, y_2, ..., y_{|Y|}\}$ é dada por :

$$EDR(X,Y) = \begin{cases} |X|, & \text{se } |Y| = 0 \\ |Y|, & \text{se } |X| = 0 \\ \text{caso contrário,} & \\ \quad min \begin{cases} EDR(Rest(X), Rest(Y)) + Dist_{edr}(x_1, y_1) \\ EDR(Rest(X), Y) + 1 \\ EDR(X, Rest(Y)) + 1 \end{cases} & \end{cases} \tag{16}$$

As vantagens da EDR em relação às outras medidas são inúmeras. Basicamente:

- Redução dos efeitos de ruídos através da quantização da distância entre um par de elementos para $\{0, 1\}$;
- Os efeitos de *outliers* são menores que na distância euclidiana, DTW e métrica ERP.

- Trata a translação temporal através das "operações de edição";
- Ao contrário da função de distância LCSS (Seção 2.1.3), atribui penalidades para extensões entre subseqüências que se equivalem, portanto neste quesito tendo mais acurácia que a LCSS.

Distância de Edição com Penalidade Real (*Edit Distance with Real Penalty*)

A métrica de similaridade ERP é levemente diferente das funções de distância DTW e EDR. A principal diferença é que a ERP é uma métrica no sentido formal, conforme definição 2. Esta propriedade possibilita a implementação de diversas estruturas de indexação [CN04].

Como a EDR, a ERP se baseia no modelo de distância de edição, generalizando a função de distância de Levenshtein da equação 4. A função de distância ERP é concebida para possuir a característica da desigualdade triangular. A principal diferença da função de distância ERP em relação à EDR é que esta utiliza o limiar ε para calcular o custo como 0 ou 1; já a ERP utiliza um valor constante g para o cálculo do custo de extensões.

O conceito de extensão é relacionado ao conceitos de alinhamento e operação de edição (Seção 2.1.2). Ao se comparar duas séries temporais X e Y, por exemplo, pode ocorrer de $|X| \neq |Y|$. Portanto, é necessário incluir ou excluir elementos em uma das séries, de forma que estas possam ser comparadas, elemento a elemento. Assim, o caractere "g" é o símbolo abstrato para extensão. Na equação 19 que segue, quando $|X| = 0$ então a distância entre $|X|$ e $|Y|$ equivale a soma das distâncias entre os símbolos de $|Y|$ - $\{y_1, ..., y_{|Y|}\}$ - e extensões, $\sum_1^{|Y|} Dist_{erp}(y_i, g)$.

A função $Dist_{erp}$ é definida para o cálculo do custo dos elementos que não se casam, dados x_i e y_i, elementos das séries X e Y, respectivamente. Esta função pode assim ser expressa:

$$Dist_{erp}(x_i, y_i) = \begin{cases} |x_i - y_i| \ , & \text{Se } x_i, y_i \text{ não são extensões} \\ |x_i - g| \ , & \text{Se } y_i \text{ é uma extensão} \\ |y_i - g| \ , & \text{Se } x_i \text{ é uma extensão} \end{cases} \tag{17}$$

O lema 1 foi formulado por [CN04]. Ele diz respeito à validade da desigualdade triangular para a função $Dist_{erp}$:

Lema 1 *Para 3 elementos quaisquer q_i, r_i, s_i, onde quaisquer um deles pode ser uma extensão, tem-se que:*

$$dist_{erp}(q_i, s_i) \leq dist_{erp}(q_i, r_i) + dist_{erp}(r_i, s_i) \tag{18}$$

A partir da definição da função de distância na equação 17, e de um valor constante $g \in \mathbb{R}$, a métrica ERP, pode ser expressa para o cálculo da distância entre duas séries temporais X e Y:

$$ERP(X,Y) = \begin{cases} \sum_1^{|X|} Dist_{erp}(x_i, g)\ , & \text{se } |Y| = 0 \\ \sum_1^{|Y|} Dist_{erp}(y_i, g)\ , & \text{se } |X| = 0 \\ \text{Caso contrário,} & min \begin{cases} ERP(Rest(X), Rest(Y)) + Dist_{erp}(x_1, y_1) \\ ERP(Rest(X), Y) + Dist_{erp}(x_1, g) \\ ERP(X, Rest(Y)) + Dist_{erp}(y_1, g) \end{cases} \end{cases} \tag{19}$$

A principal diferença da métrica ERP em relação às outras métricas, é a validade da desigualdade triangular. Com respeito à recuperação de informações e à execução de consultas, algumas estruturas interessantes de indexação poderiam ser utilizadas como a Slim-Tree[TTSF00]. O teorema 1 que se segue, expressa esta característica importante:

Teorema 1 *(Desigualdade Triangular) Sejam Q, R, S três séries temporais de tamanhos arbitrários. Então tem-se que : $ERP(Q,S) \leq ERP(Q,R) + ERP(R,S)$.*

Maior Subseqüência Comum para Sinais (*Longest Common Subsequence*)

A função de distância LCSS é definida para o domínio de séries temporais seguindo o modelo da distância de edição. Na sua concepção considera-se que uma função de distância prática deve tratar seqüências com ruídos,

tamanhos diferentes, *outliers*, diferentes velocidades ou taxas de amostragem, translações no tempo (no caso de séries temporais), tudo isso de uma forma eficiente. A LCSS atende estes requisitos, considerando alguns ajustes [Gun02].

A função LCSS utiliza um limiar ε que define se dois elementos da série temporal se equivalem ou não. Se os elementos se equivalem eles somam 1 na medida LCSS. Caso contrário, o restante da série, sem os elementos, é analisada. Para comparar elementos das duas séries temporais X e Y, a função de distância LCSS é expressa[CN04] através da recorrência:

$$LCSS(X,Y) = \begin{cases} 0 & \text{, se } |Y| = 0 \text{ ou } |X| = 0 \\ 1 + LCSS(Rest(X), Rest(Y)) & \text{, se } Dist(x_1, y_1) \leq \varepsilon \\ max \begin{cases} LCSS(Rest(X), Y) \\ LCSS(X, Rest(Y)) \end{cases} & \text{, caso contrário} \end{cases} \tag{20}$$

A medida LCSS é bem robusta, no entanto, a LCSS não obedece à desigualdade triangular como a métrica ERP.

Sumário para cálculo das Funções de distância

A tabela 2.1 apresenta um resumo das fórmulas de cálculo das funções de distância, em sinais digitais.

2.2 Indexação Métrica

A técnica de indexação métrica foi utilizada em dois momentos nesta pesquisa. O primeiro uso foi na etapa de conversão simbólica, mais precisamente no algoritmo de clusterização PAM-SLIM[BRTJ06]. O segundo uso foi na etapa de reconhecimento, acelerando a execução do classificador kNN.

A indexação métrica tem o propósito de recuperar eficientemente objetos

Sumário para cálculo de funções, em sinais digitais

$$Dist_{L_1}(x,y) = \sum_{i=1}^{n} |x[i] - y[i]|, Dist_{L_2}(x,y) = \sqrt{\sum_{i=1}^{n}(x[i] - y[i])^2}$$

$$Dist(x,y) = Dist_{L_1}(x,y) \text{ ou } Dist_{L_2}(x,y)$$

$$Rest(X) = x_2, ..., x_{|X|}$$

$$Dist_{edr}(x_i, y_i) = \begin{cases} 0 & \text{Se } |x_i - y_i| \leq \varepsilon \\ 1 & \text{Caso contrário} \end{cases}$$

$$DTW(X,Y) = \begin{cases} 0 & \text{, se } |X| = |Y| = 0 \\ \infty & \text{, se } |X| = 0 \text{ ou } |Y| = 0 \\ Dist(x_1, y_1) + min \begin{cases} DTW(Rest(X), Rest(Y)) \\ DTW(Rest(X), Y) \\ DTW(X, Rest(Y)) \end{cases} & \text{, caso contrário} \end{cases}$$

$$EDR(X,Y) = \begin{cases} |X| & \text{se } |Y| = 0 \\ |Y| & \text{se } |X| = 0 \\ \text{caso contrário,} & min \begin{cases} EDR(Rest(X), Rest(Y)) + Dist_{edr}(x_1, y_1) \\ EDR(Rest(X), Y) + 1 \\ EDR(X, Rest(Y)) + 1 \end{cases} \end{cases}$$

$$ERP(X,Y) = \begin{cases} \sum_{1}^{|X|} Dist_{erp}(x_i, g) & \text{se } |Y| = 0 \\ \sum_{1}^{|Y|} Dist_{erp}(y_i, g) & \text{se } |X| = 0 \\ \text{Caso contrário,} & min \begin{cases} ERP(Rest(X), Rest(Y)) + Dist_{erp}(x_1, y_1) \\ ERP(Rest(X), Y) + Dist_{erp}(x_1, g) \\ ERP(X, Rest(Y)) + Dist_{erp}(y_1, g) \end{cases} \end{cases}$$

$$LCSS(X,Y) = \begin{cases} 0 & \text{se } |Y| = 0 \text{ ou } |X| = 0 \\ 1 + LCSS(Rest(X), Rest(Y)) & \text{se } Dist(x_1, y_1) \leq \varepsilon \\ \text{Caso contrário, } max \begin{cases} LCSS(Rest(X), Y) \\ LCSS(X, Rest(Y)) \end{cases} & \end{cases}$$

Tabela 2.1: Sumário para cálculo de funções, em sinais digitais

similares a outro objeto de consulta fornecido. Para a definição de um espaço métrico é necessário a definição de um domínio e uma métrica[Ron97].

Esta técnica lida com consulta em espaços métricos. As consultas mais comumente usadas são a consulta por abrangência - *Range Query* - e a consulta por vizinhos mais próximos - *k-Nearest Neighbor Query* - [KSF+96], formalizadas nas definições 4 e 5.

Definição 4 *Consulta por Abrangência (Range Query) : Dado um objeto de consulta $Q \in \mathrm{D}$ e uma distância máxima $r(Q)$, a consulta por abrangência $(Q, r(Q))$ seleciona todos os objetos indexados O_j tal que $d(O_j, Q) \leq r(Q)$.*

Definição 5 *Consulta por vizinhos mais próximos (k nearest neighbors (kNN) query) : Dado um objeto de consulta $Q \in \mathrm{D}$ e um inteiro $k \geq 1$, a consulta kNN (Q, k) seleciona os k objetos indexados que têm a menor distância de Q, de acordo com uma função de distância d.*

Dessa forma, uma estrutura de indexação métrica, ou ainda Método de Acesso Métrico (MAM), otimiza a execução de consultas por similaridade. A MAM utilizada neste trabalho foi a Slim-Tree[TTSF00] que é uma melhoria da M-Tree [CPRZ97]

2.2.1 M-Tree

A estrutura M-Tree [CPRZ97] organiza objetos $O_i \in \mathrm{D}$ em nós fixos, que correspondem às regiões de um espaço métrico $< \mathrm{D}, d >$, onde d é uma métrica e D um domínio de dados. Os nós de uma M-Tree podem armazenar até M entradas - esta é a capacidade dos nós. Os nós podem ser nós folha ou nós internos. Cada uma das M entradas em um nó folha é do tipo:

$$entry(O_j) = [O_j, oid(O_j), d(O_j, P(O_j))] \tag{21}$$

A entrada da equação 21 é armazenada em um nó folha. O identificador $oid(O_j)$ é uma referência ao objeto indexado, que reside em um arquivo separado; O_j é um vetor de parâmetros do objeto, em outras palavras, $O_j \in \mathrm{D}$; também $d(O_j, P(O_j))$ é a distância entre O_j e um objeto pai, $P(O_j)$.

Uma entrada em um nó interno (não folha) armazena um vetor de parâmetros - O_τ - também chamado objeto de roteamento, e um raio de cobertura, $r(O_\tau) > 0$. Uma entrada para um objeto de roteamento O_τ inclui um ponteiro - $ptr(T(O_\tau))$ - para a raiz da sub-árvore $T(O_\tau)$ - a árvore de cobertura de O_τ - e $d(O_\tau, P(O_\tau))$, a distância de O_τ para seu pai:

$$entry(O_\tau) = [O_\tau, ptr(T(O_\tau)), d(O_\tau, P(O_\tau))] \quad (22)$$

A semântica do raio de cobertura é estruturada a partir da propriedade seguinte:

Propriedade 1 *O raio de cobertura de um objeto de roteamento, O_τ, satisfaz a desigualdade $d(O_j, O_\tau) \leq r(O_\tau)$, para cada objeto O_j armazenado na árvore de cobertura de O_τ.*

A propriedade 1 significa que um objeto de roteamento define uma região no espaço métrico M, centralizado em O_τ e com raio $r(O_\tau)$, como na Figura 2.2. Além disso O_τ é pai de cada objeto O_j armazenado na árvore referenciada por $ptr(T(O_\tau))$. A implicação dessa organização é que uma M-Tree organiza um espaço métrico em um conjunto de regiões que podem se sobrepor, nas quais recursivamente podem ser definidas outras regiões que indexam o espaço métrico.

Como a M-Tree cresce

Como qualquer outra árvore balanceada dinâmica, a M-Tree cresce de uma maneira *bottom-up*. O *overflow* de um nó N é gerenciado alocando-se um novo nó, N', de mesmo nível de N, particionando as $M + 1$ entradas entre dois nós, e então postando informações relevantes para o nó pai N_p. Quando uma raiz é particionada, uma nova raiz é criada e a M-Tree cresce um nível, veja a Figura 2.3. Este método é denominado particionamento (*Split*) - veja o algoritmo 1.

No algoritmo *Split* existem dois métodos importantes, denominados *Partition e Promote*. O método *Partition* particiona as $(M + 1)$ entradas de um nó cheio em dois subconjuntos disjuntos, N_1 e N_2, que são então armazenados em nós N e N', respectivamente. Uma implementação específica do método *Promote* e *Partition* define o que se chama de *split policy*. A estrutura M-Tree oferece a possibilidade de implementação de diferentes *split policies*, que podem ser otimizadas de acordo com as necessidades da aplicação. Cada *split policy* tem que respeitar a semântica de cobertura do raio, conforme definido na propriedade 1. Se um nó cheio N é uma folha, isto é garantido fazendo-se:

Algorithm 1 Split

Entrada: Um nó N de árvore métrica T, uma entrada E.
Saída: O nó N é particionado em dois nós N_1 e N_2
Método: O algoritmo funciona assim:

```
Seja η = entradas do nó N ∪ {E};
if N não é a raiz then
    Seja O_p o pai de N, armazenado em N_p;
end if
Crie umn novo nó N';
Promote (η,O_p1,O_p2);
Partition (η,O_p1,O_p2,η_1,η_2);
Armazene as entradas de η_1 em N e as entradas de η_2 em N';
if N é a raiz atual then
    Crie um novo nó, N_p;
    Armazene entry(O_p1) e entry(O_p2) em N_p;
else
    Substitua entry(O_p) com entry(O_p1) em N_p;
    if O nó N_p está cheio then
        Split (N_p,entry(O_p2))
    else
        Armazene entry(O_p2) em N_p;
    end if
end if
```

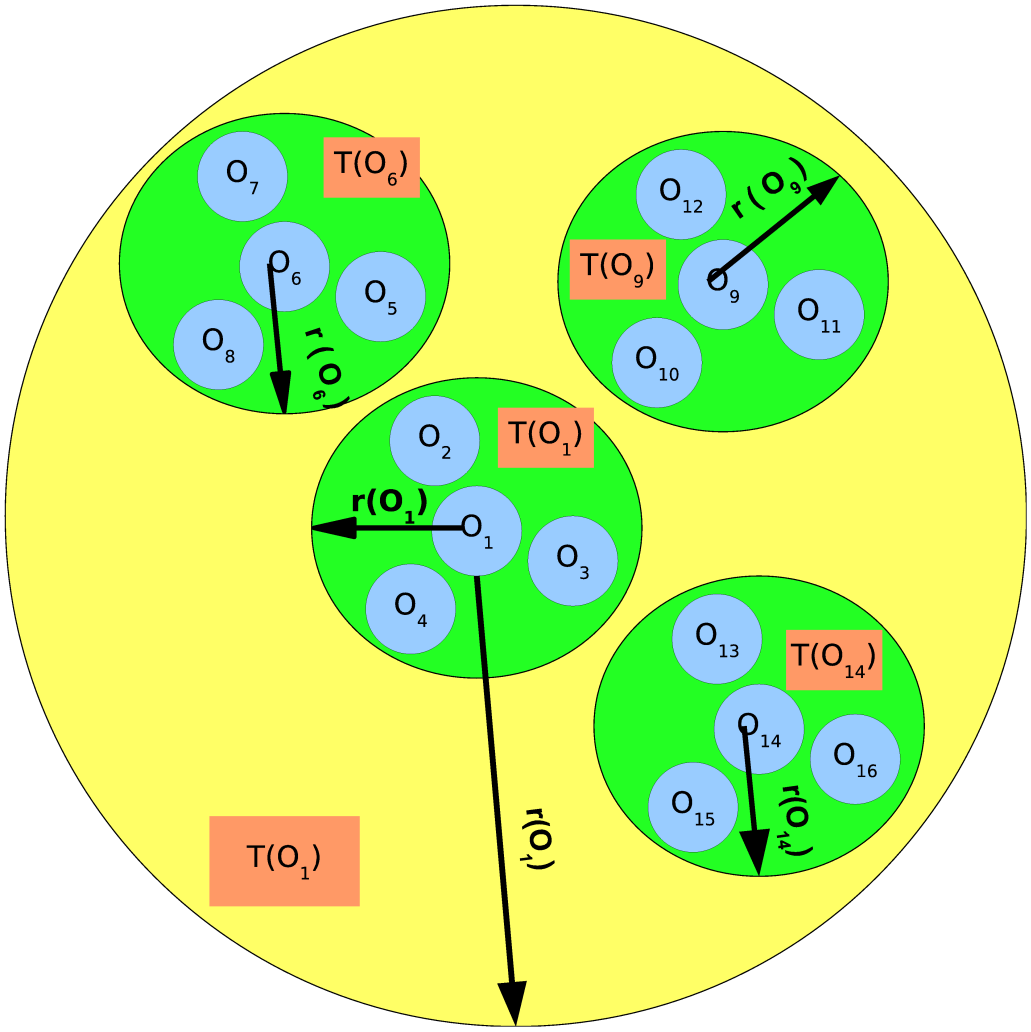


Figura 2.2: Representação de uma região do espaço métrico, a partir de um objeto de roteamento

$$r(O_{p1}) = max\{d(O_j, O_{p1}) | O_j \in \mathrm{N}_1\} \tag{23}$$

De uma forma geral, o raio de cobertura de um objeto de roteamento apontando para uma folha é igual à distância máxima entre o objeto de roteamento e os objetos na folha. Quando o *split* envolve nós internos, N, cada entrada O_j em N_1 tem um raio não nulo de cobertura, $r(O_j)$. Assim tem-se:

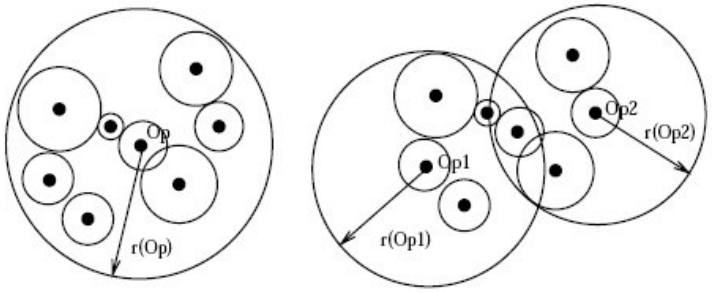


Figura 2.3: Particionamento de uma M-Tree

$$r(O_{p1}) = max\{d(O_j, O_{p1}) + r(O_j) | O_j \in \mathrm{N}_1\} \tag{24}$$

Logo, a equação 24 é garantida pela propriedade de desigualdade triangular, isto é, nenhum objeto em $T(O_{p1})$ pode se distanciar de O_{p1} mais que $r(O_{p1})$. Um exemplo é a Figura 2.3. O procedimento de inserção é apresentado no algoritmo 6.

Processando consultas de similaridade

A estrutura M-Tree tem como objetivo reduzir o tempo processamento nas consultas por similaridade. Para tal, os algoritmos desenvolvidos em [CPRZ97] diminuem sensivelmente o número de cálculos de distância, utilizando-se de distâncias pré-calculadas armazenadas.

Consultas de Abrangência

Uma consulta por abrangência seleciona todos os objetos armazenados no banco de dados tal que $d(O_j, Q) \leq r(Q)$. Em outras palavras, o interesse é selecionar todos os objetos que se distanciam de Q por uma distância de até $r(Q)$. Para executar eficientemente esta consulta, uma condição importante é aplicada, conforme segue.

Lema 2 *Se $d(O_\tau, Q) > r(Q) + r(O_\tau)$, então para cada objeto O_j em $T(O_\tau)$*

tem-se que $d(O_j, Q) > r(Q)$. *Portanto,* $T(O_\tau)$ *pode ser seguramente ignorado na busca.*

De fato, uma vez que $d(O_j, Q) \geq d(O_\tau, Q) - d(O_j, O_\tau)$ (pela desigualdade triangular) e $d(O_j, O_\tau) \leq r(O_\tau)$ (pela definição de raio de cobertura), daí tem-se que $d(O_j, Q) \geq d(O_\tau, Q) - r(Q)$. Em outras palavras $d(O_\tau, Q) - r(Q) > r(Q)$.

Lema 3 *Se* $|d(O_p, Q) - d(O_\tau, O_p)| > r(Q) + r(O_\tau)$, *então* $d(O_\tau, Q) > r(Q) + r(O_\tau)$.

A consulta por abrangência foi pode ser realizada através do Algoritmo 2 (RS).

Consulta de vizinhos mais próximos

A consulta kNN recupera os k vizinhos mais próximos de um objeto de consulta Q. Assume-se que pelo menos k objetos são indexados por uma M-Tree. Utiliza-se uma técnica *branch-and-bound* a qual utiliza duas estruturas globais: uma fila de prioridades PR e um um vetor de k elementos, NN que irá conter o resultado no final de execução.

PR é uma fila de ponteiros para ativar sub-árvores, isto é, sub-árvores onde os objetos qualificados podem ser encontrados. Com o ponteiro para raiz da sub-árvore $T(O_\tau)$, um limite inferior $d_{min}(T(O_\tau))$ na distância de qualquer objeto em $T(O_\tau)$ a Q é tomado. O limite inferior é:

$$d_{min}(T(O_\tau)) = max\{d(O_\tau, Q) - r(O_\tau), 0\} \tag{25}$$

Pela equação 25, nenhuma $T(O_\tau)$ pode ter uma distância até Q menor que $d(O_\tau, Q) - r(O_\tau)$. Estes limites são usados pela função ChooseNode - algoritmo 3 - para extrair de PR o próximo nó a ser examinado.

O critério de otimização é dinâmico, uma vez que o raio de busca é a distância entre Q os k-vizinhos mais próximos correntes. Também a ordem

Algorithm 2 RS

Entrada: Um nó N de árvore métrica T, um objeto de consulta Q, e um raio de busca $r(Q)$.

Saída: São retornados em $result$ uma lista de objetos O_j tal que $d(O_j, Q) \leq r(Q)$

Método: O algoritmo funciona assim:

```
Seja O_p o pai do nó N;
if N é um nó interno then
  for ∀O_τ ∈ N faça: do
    if |d(O_p, Q) − d(O_τ, O_p)| ≤ r(Q) + r(O_τ) then
      Calcule d(O_τ, Q);
      if d(O_τ, Q) ≤ r(Q) + r(O_τ) then
        RS (∗ptr(T(O_τ)),Q,r(Q));
      end if
    end if
  end for
else
  for ∀O_j ∈ N faça: do
    if |d(O_p, Q) − d(O_j, O_p)| ≤ r(Q) then
      Calcule d(O_j, Q);
      if d(O_j, Q) ≤ r(Q) then
        Adicione oid(O_j) em result;
      end if
    end if
  end for
end if
```

em que os nós são visitados pode afetar a performance. Um critério heurístico já implementado em [CPRZ97] é selecionar o nó em que o limite d_{min} é mínimo.

Algorithm 3 ChooseNode

Entrada: Uma fila de prioridades PR
Saída: É retornado o próximo nó a ser examinado
Método: O algoritmo funciona assim:
Seja $d_{min}(T(O_\tau^*)) \leftarrow min\{d_{min}(T(O\tau))\}$, considerando todas as entradas em PR;
Remova a entrada $[ptr(T(O_\tau^*)), dmin(T(O_\tau^*))]$ de PR;
Retorne $*ptr(T(O_\tau^*))$;

Algorithm 4 kNN_Search

Entrada: Um nó raíz T, um objeto de consulta Q, e um inteiro k.
Saída: Os k objetos mais próximos são retornados através do vetor NN
Método: O algoritmo funciona assim:
$PR = [T, -]$
for $i = 1$ até k faça: **do**
$NN[i] = [-, \infty]$
end for
while $PR \neq \emptyset$ faça: **do**
$NextNode = ChooseNode(PR)$;
$kNN_NodeSearh(NextNode, Q, k)$;
end while

No final da execução, a i-ésima entrada do array NN terá o valor $NN[i] = [oid(O_j), d(O_j, Q)]$, com O_j sendo o i-ésimo vizinho mais próximo de Q. O valor da distância da i-ésima entrada é denotado d_i de forma que d_k é o a distância mais larga em NN. Claramente, d_k faz o papel de raio dinâmico de busca, desde que qualquer sub-árvore em que $d_{min}(T(O_\tau)) > d_k$ pode ser ignorada com segurança.

As entradas do array NN são inicialmente configuradas para $NN[i] = [-, \infty](i = 1, ..., k)$, isto é, os oids são indefinidos e $d_i = \infty$. Quando a busca é iniciada e os nós inteiros são acessados, a idéia é então calcular, para cada sub-árvore $T(O_\tau)$, um limite superior, $d_{max}(T(O_\tau))$, na distância de qualquer objeto em $T(O_\tau)$ para Q. O limite superior é configurado para

$$d_{max}(T(O_\tau)) = d(O_\tau, Q) + r(O_\tau) \tag{26}$$

Algorithm 5 kNN_NodeSearch

Entrada: Um nó N , um objeto de consulta Q, e um inteiro k.
Saída: É retornado um conjunto melhor no array NN
Método: O algoritmo funciona assim:

```
Seja O_p o pai do nó N;
if N é um nó interno then
  for ∀O_τ ∈ N faça: do
    if |d(O_p, Q) − d(O_τ, O_p)| ≤ d_k + r(O_τ) then
      Calcule d(O_τ, Q);
      if d_min(T(O_τ)) ≤ d_k then
        Adicione [ptr(T(O_τ)), d_min(T(O_τ))] em PR;
        if d_max(T(O_τ)) < d_k then
          d_k ← NN_Update([−, dmax(T(O_τ))]);
          Remova de PR todas as entradas onde d_min(T(O_τ)) > d_k;
        end if
      end if
    end if
  end for
else
  for ∀O_j ∈ N faça: do
    if |d(O_p, Q) − d(O_j, O_p)| ≤ d_k then
      Calcule d(O_j, Q);
      if d(O_j, Q) ≤ d_k then
        d_k ← NN_Update([oid(O_j), d(O_j, Q)]);
        Remova de PR todas as entradas onde d_min(T(O_τ)) > d_k;
      end if
    end if
  end for
end if
```

Considere o caso mais simples onde $k = 1$, duas sub-árvores, $T(O_{r1})$ e $T(O_{r2})$, e assuma que $dmax(T(O_{r1})) = 5$ e $d_{min}(T(O_{r2})) = 7$. Uma vez que $d_{max}(T(O_{r1})$ garante que um objeto cuja distância de Q é de pelo menos 5 existe em $T(O_{r1})$,$T(O_{r2})$, então as mesmas podem ser ignoradas na busca. O método kNN_Search é definido pelo algoritmo 4. Outro método utilizado é o método kNN_NodeSearch - algoritmo 5.

Inserindo objetos

Para inserir um novo nó O_n em uma estrutura M-Tree tenta-se achar um nó O_j cujo raio de cobertura seja maior que $d(O_j, O_n)$; se somente um nó é encontrado então o novo nó é inserido neste nó interno, caso hajam vários nós internos candidatos, o novo nó é inserido naquele que têm a menor distância.

No caso de não existirem nós cujo raio de cobertura seja compatível com o novo nó, seleciona-se o nó interno com menor distância e então se aumenta o raio de cobertura deste nó após a inserção. Se o nó está cheio - atingiu M filhos - então realiza-se o procedimento *split* já descrito anteriormente. O algoritmo 6 define formalmente um método de inserção na M-Tree.

2.2.2 Slim-Tree

A SlimTree é igual a M-Tree [Sko04], do ponto de vista estrutural. Do ponto de vista da análise de algoritmos, a inovação são as otimizações na estrutura hierárquica, o que promove um grande aumento na performance. O algoritmo *slim-down* 7 otimiza a estrutura hierárquica já estabelecida através de inserções de objetos.

A idéia básica é diminuir a sobreposição de áreas de cobertura em nós não folha. Dado uma entrada em um nó folha, assume-se que existe uma folha mais adequada para esta entrada. Seja uma entrada $entry(O_j)$, armazenada em uma folha N (tendo como pai o objeto de roteamento O_τ), tal que $d(O_j, O_\tau) < r(O_\tau)$. Se tal folha N' existe (mais adequada que N), a entrada $entry(O_j)$ é então inserida nela (sem a necessidade de alargar N'), e removida de N com a diminuição do raio de cobertura $r(O_\tau)$ (Se O_j era o objeto mais distante de O_j em N). O procedimento é repetido para todas as entradas em todas as folhas à medida que o movimento de entradas ocorre.

Algorithm 6 Insert

Entrada: Um nó N de árvore métrica T, uma entrada $entry(O_n)$.
Saída: Insere o objeto O_n na folha mais apropriada.
Método: O algoritmo funciona assim:

```
if N é um nó interno then
  Seja N_in = entradas onde d(O_τ, O_n) ≤ r(O_τ);
  if N_in ≠ ∅ then
    Seja entry(O*_τ) ∈ N_in : d(O*_τ, O_n) é mínima;
  else
    Seja entry(O*_τ) ∈ N_in : d(O*_τ, O_n) − r(O*_τ) é mínima;
    Seja r(O*_τ) = d(O*_τ, O_n);
  end if
  Insert (∗ptr(T(O*_τ)), entry(O_n));
else
  if N não está cheia then
    Armazene entry(O_n) em N
  else
    Split (N,entry(O_n));
  end if
end if
```

A aplicação da versão original do algoritmo *slim-down* otimiza a performance das buscas em 35% (em média)[TTSF00].

Quantificação da sobreposição entre áreas de cobertura

A noção de volume não está disponível em um espaço métrico; Já a noção de sobreposição entre áreas de cobertura em uma estrutura Slim-Tree pode ser medida através de uma abordagem diferente: A contagem de objetos que estão em ambas áreas, onde se deseja calcular a intensidade da sobreposição[TTSF00]- veja a definição 6.

Definição 6 *Sejam I_1 e I_2 duas entradas indexadas. A sobreposição entre I_1 e I_2 é definida como o número de objetos nas sub-árvores correspondentes que são cobertos por ambas as regiões, dividido pelo número de objetos em ambas as árvores.*

A definição 6 apresenta uma maneira genérica de medir a intersecção entre regiões de uma árvore métrica, habilitando o uso de técnicas de otimização, desenvolvidas para espaços vetoriais, em árvores métricas.

Fat Factor

O *fat factor* mede o quão adequado está uma árvore métrica quanto à sobreposição entre suas áreas de cobertura. Uma árvore ideal não possui sobreposição de áreas; assim o *fat factor* tem valor 0. A árvore mais indesejável quanto à sobreposição possui *fat factor* no valor de 1 - veja a definição 7.

Definição 7 *Seja T uma árvore métrica com altura H e M nós, $M \geq 1$. Seja N o número de objetos. Então o fat factor de uma árvore métrica T é:*

$$fat(T) = \frac{I_c - H * N}{N} \times \frac{1}{(M - H)} \tag{27}$$

Onde I_c denota o número total de nós acessados para responder a consulta a cada um dos N objetos armazenados em uma árvore métrica.

Bloat Factor

O *bloat factor*, apresentado na definição 8, possibilita a comparação de duas árvores, considerando a quantidade de sobreposição e ainda a ocupação eficiente dos nós. A idéia é "penalizar"árvores que usam mais nós que o número mínimo requerido para indexar uma coleção de objetos. Entre todas as possíveis árvores, a árvore mínima é aquela com altura mínima H_{min} e o número mínimo de nós M_{min}.

Definição 8 *O bloat factor de uma árvore métrica T com mais que um nó é expresso como:*

$$bl(T) = \frac{I_C - hmin * N}{N} \times \frac{1}{(M_{min} - H_{min})} \tag{28}$$

O *bloat factor* da definição 8 varia entre 0 e um número positivo que pode ser maior que 1. Este fator possibilita a comparação direta entre árvores; Árvores com menor *bloat factor* sempre terão um menor número de acessos a disco, na execução de consultas.

O algoritmo Slim-down

O algoritmo 7, denominado *Slim-down*, produz uma árvore métrica mais compacta, com redução das medidas *bloat-factor* e *fat-factor*[TTSF00]. A Figura 2.4 exemplifica o funcionamento do algoritmo *Slim-down*, onde um objeto c muda de um nó i para um nó j, ocorrendo, portanto uma redução no raio de cobertura de i. O número de iterações do *laço* mais externo do algoritmo é limitado a 3; isto ocorre porque existem casos em que podem ocorrer movimentos cíclicos entre nós, à medida que as iterações ocorrem.

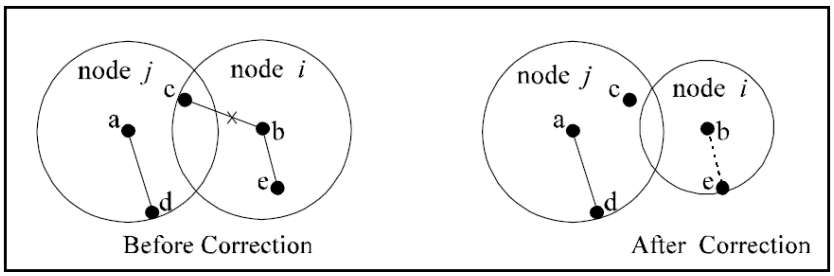


Figura 2.4: Funcionamento do algoritmo Slim-Down

Política de particionamento na Slim-Tree

Em [TTSF00] são definidos várias políticas de particionamento dessa árvore métrica. As estratégias adotadas são:

- Randômica: Dois nós centrais são randomicamente selecionados e os objetos existentes são distribuídos entre eles. Cada objeto é armazenado em um novo nó que tem seu centro mais próximo deste objeto. Esta estratégia é bem rápida.
- MinMax: Todos os possíveis pares de objetos são considerados com representantes potenciais. Para cada par, um algoritmo linear atribui os

Algorithm 7 Slim down

Entrada: Uma árvore métrica T.
Saída: A árvore T pode as medidas *bloat-factor* e *fat-factor* reduzidas;
Método: O algoritmo funciona assim:

```
passos ← 1
mudanças ← 1
while mudanças = 1 E passos ≤ 3 do
  mudanças ← 0
  for cada nó i ∈ T do
    Encontre o objeto c mais distante do representante b de i
    if ∃j (não vazio que também cobre c) then
      Remova c de i e insira ele em j.
      Corrija o raio do nó i
      mudanças ← 1
    end if
  end for
  passos ← passos + 1
end while
```

objetos para um representante. O par que minimiza o raio de cobertura é escolhido. A complexidade do algoritmo é $\Theta(C^3)$, usando $\Theta(C^2)$ para cálculos de distância.

- MST: A árvore de caminho mínimo dos objetos é gerada, e aresta mais longa é removida.

A estratégia de particionamento baseada na árvore de caminho mínimo - algoritmo 8 - tem se mostrado muito robusta; portanto, a mesma política padrão de particionamento é utilizada na estrutura Slim-Tree.

2.3 Clusterização via Indexação métrica e k-medianas

Clusterização é o processo de dividir dados ou objetos em grupos, de forma que os componentes de um determinado grupo, ou cluster, sejam mais similares aos objetos dentro do grupo que com os objetos fora do grupo [Yu05].

Algorithm 8 Estratégia de particionamento baseada na árvore de caminho mínimo

Entrada: Um nó N de árvore métrica T.
Saída: Dois nós N_1, N_2 são obtidos a partir de N
Método: O algoritmo funciona assim:
Construa a MST dos C objetos em N
Remova a aresta mais longa
Transforme os dois grupos resultantes nos nós N_1 e N_2
Escolha o representante de cada grupo minimizando-se a distância máxima entre o representante e os outros elementos do grupo.

No contexto deste trabalho, a técnica de clusterização é de fundamental importância, uma vez que a QBER tem como pré-requisito a utilização de algum algoritmo de clusterização.

Existem vários tipos de clusterização, baseando-se em diversas visões e teorias sobre o mesmo problema. Em geral, tais soluções podem ser agrupadas em dois grandes grupos: algoritmos de particionamento e algoritmos hierárquicos. Os algoritmos hierárquicos produzem uma hierarquia de clusters e níveis; já os algoritmos de particionamento dividem os dados em k grupos, onde k é fornecido previamente. Os algoritmos k-médias (*k-means*) e k-medianas (*k-medoids*) são exemplos de algoritmos de particionamento.

Este trabalho utiliza o algoritmo k-medianas. A despeito de todas as vantagens do k-médias, como desempenho e simplicidade, existem alguns problemas identificados. O k-médias supõe a utilização da operação de média de objetos, possível em espaços vetoriais. A operação de média pode provocar algumas distorções indesejáveis. Os objetos que se têm interesse clusterizar, são objetos de um espaço métrico, considerando a medida de similaridade ERP, onde não há a operação de média definida. O algoritmo k-medianas poderia evitar distorções indesejáveis, em espaços métricos onde a operação de média não é desejável ou viável.

O grande problema do algoritmo k-medianas, em sua versão exata é o grande esforço computacional necessário para geração dos clusters. Muitos pesquisadores propuseram soluções aproximadas, considerando-se os paradigmas da amostragem estatística e peculiaridades das funções de similaridade utilizadas.

2.3.1 Algoritmos que Implementam o K-medianas

A abordagem k-medianas é simples. Dado um conjunto de n objetos $O = \{o_1, o_2, ..., o_n\}$, o objetivo é encontrar k clusters, representados cada um por sua centróide (elemento mais representativo do cluster). A abordagem em questão produz um conjunto $C = \{c_1, c_2, ..., c_k\}$, onde c_i é a centróide do cluster i. Cada objeto do conjunto O está no cluster cuja centróide é mais similar ao objeto. De um modo geral, os algoritmos que implementam a abordagem k-medianas tem os seguintes passos genéricos:

- Inicialização: Dado um conjunto de objetos, selecione, de alguma forma, k objetos como centróides;
- Avaliação: O objetivo é encontrar uma solução C que minimize a função objetiva:

$$\sum_{j=1}^{n} d(o_j, c_i), \tag{29}$$

  onde c_i é a centróide do cluster atribuído a o_j.

Existem 3 algoritmos bem conhecidos baseados na abordagem k-medianas:

Partitioning Around Medoids (PAM)

O algoritmo PAM começa selecionando arbitrariamente k centróides. O passo seguinte é encontrar um objeto associado a um cluster, que pode substituir sua respectiva centróide, isto é, minimizar a função objetiva da equação 29. Se tal objeto é encontrado, os objetos têm suas centróides reatribuídas e, em seguida realiza-se o procedimento novamente; Se não é encontrado um objeto que diminui o valor da função objetiva o algoritmo para.

Clustering LARge Applications (CLARA)

O algoritmo CLARA aprimora o PAM para o uso em grande quantidade de dados. Utiliza-se amostragem para diminuir a complexidade computacional do k-medianas. O algoritmo trabalha da seguinte forma: seleciona-se q

amostras aleatórias de um conjunto de dados. Em seguida, executa-se o PAM para cada uma das q amostras, calculando-se um conjunto de centróides para cada amostra. O último passo é verificar qual conjunto de centróides miniza a função objetiva da equação 29, considerando todo conjunto de dados. Experimentos dos autores mostram que 5 amostras contendo $40 + 2k$ objetos são suficientes para obtenção de resultados satisfatórios.

CLARANS

O desenvolvimento do algoritmo CLARANS[NH94] teve como objetivo a melhora de performance do algoritmo CLARA, juntamente com a melhoria na qualidade dos cluster produzidos. A técnica utilizada é a busca aleatória no espaço de soluções possíveis. Ao invés de amostrar o espaço de objetos e, em seguida, rodar o PAM nestas amostras de objetos, o CLARANS trabalha sob amostras das soluções possíveis.

O algoritmo em questão utiliza-se do conceito de nós vizinhos. Cada solução possível é um nó em um grafo imaginário. Dois nós são vizinhos se as soluções que os mesmos representam (conjuntos de centróides) diferem somente por um elemento. São amostradas as soluções possíveis e os vizinhos dessas soluções. Os parâmetros do algoritmo seriam o número máximo de vizinhos verificados e o número máximo de soluções arbitrárias verificadas.

Em relação aos algoritmos anteriores PAM e CLARA, o CLARANS apresenta a melhor relação custo benefício entre tempo de processamento e qualidade dos clusters gerados.

2.3.2 Otimização do K-medianas via Estruturas de Indexação Métrica

Os algoritmos apresentados nas seções anteriores são algoritmos gerais para a clusterização via k-medianas. De acordo com o tipo de função de similaridade empregado, podem ser feitas outras otimizações. A função de distância empregada na QBER, em nossa implementação de referência, é a métrica ERP. A característica métrica da ERP propicia inúmeras otimizações, entre elas a possibilidade de utilização de estruturas de indexação métrica.

A partir do emprego de estruturas de indexação métrica, os algoritmos da abordagem k-medianas podem ser otimizados, considerando o emprego de métricas. Dessa forma surgiu o algoritmo PAM-SLIM [BRTJ06], que emprega a estrutura de indexação métrica Slim-Tree [TTSF00].

O algoritmo PAM-SLIM

O algoritmo PAM-SLIM utiliza propriedades da estrutura de indexação métrica Slim-Tree[TTSF00] para otimizar a execução da clusterização k-medianas. Experimentalmente, o algoritmo em questão apresenta performance comparável ao algoritmo CLARA, com qualidade de clusters gerados superior a este.

A indexação de espaços métricos tem o propósito de recuperar eficientemente objetos similares a um determinado objeto de consulta fornecido. Já a estrutura Slim-Tree é uma árvore n-ária (um pai tem n-filhos) que divide o espaço métrico em regiões. Analogamente à árvore binária, a Slim-Tree têm vários níveis; o nível central da árvore, pelas propriedades da estrutura, é uma boa amostra de toda a população que se deseja clusterizar. O assunto indexação métrica e a estrutura Slim-Tree são apresentados detalhadamente na seção 2.2.

O PAM-SLIM supõe que a execução do algoritmo PAM, no nível médio da Slim-Tree gerada, a partir dos objetos que se deseja clusterizar, seria satisfatória para encontrar k clusters de boa qualidade. Definem-se clusters de boa qualidade como clusters cuja centróides se desviam minimamente do conjunto encontrado, na execução do algoritmo PAM original. O PAM-SLIM pode ser resumido nos seguintes passos [BRTJ06]:

1. Construa a Slim-Tree para os dados a serem clusterizados;
2. Encontre o nível correspondente à metade da altura da árvore Slim-Tree. Se a quantidade de elementos neste nível for inferior ao valor de k, selecione o nível seguinte e assim por diante;
3. Clusterize os elementos do nível selecionado, utilizando o algoritmo PAM;
4. Associe cada objeto, em todo o conjunto de dados, a uma centróide retornada pelo passo anterior;

5. Retorne as centróides obtidas e também os objetos associados a cada centróide.

A eficiência do algoritmo PAM-SLIM está estritamente ligada à eficiência da estrutura de indexação métrica Slim-Tree, e à sua capacidade de dividir o espaço métrico em regiões relevantes. Tal eficácia pode ser comprovada pelo autor, em seus experimentos, e ainda por extensivos testes realizados pela criadora do PAM-SLIM[BRTJ06].

Capítulo 3

Quantização Baseada em Extremos Relativos (QBER) e a Teoria dos Extremos Relativos

Neste capítulo apresentar-se-á uma evolução da TEI[Gan03], a Teoria de Extremos Relativos (TER). Tal teoria é altamente relevante para o modelo desenvolvido nesta pesquisa. Os conceitos acrescentados são o conceitos prevalência, montes, representação RBER, representação RBERQ e finalmente a técnica QBER.

Basicamente, as teorias desenvolvidas sobre o assunto levam em consideração um simples fato originário da matemática básica: uma função matemática pode ser esboçada através do conhecimento de seus pontos críticos, interseções com os eixos coordenados, suas regiões de crescimento, suas regiões de decrescimento, seus máximos e mínimos (inclusive os locais) e seu comportamento em valores altos [Lan68].

Assim, ao se obter os valores dos extremos globais e locais para uma função e suas derivadas, é fácil esboçar seu gráfico, ou ainda, aproximar seus valores. Uma forma de simplificar a análise de uma função portanto é fazer esta análise através de seus extremos globais e locais.

3.1 Extremos Relativos

Uma série temporal pode ser analisada a partir de alguns pontos notáveis que "resumem"o seu conteúdo, assim como podemos esboçar qualquer função a partir da análise dos seus máximos e mínimos globais e locais, além dos máximos e mínimos de suas derivadas.

Os pontos notáveis mencionados podem ser denominados extremos, seguindo a terminologia de [Gan03], e são definidos como segue:

Definição 9 *Seja* $X = x_1, x_2, ..., x_n$ *uma série temporal. Seja também* x_i *um elemento desta série temporal, com* $1 < i < n$*. Então:*

- x_i *é um mínimo estrito se* $x_{i-1} < x_i$ *e* $x_i < x_{i+1}$*;*
- x_i *é um mínimo esquerdo se* $x_i < x_{i-1}$ *e ainda* $\exists k > i$ *tal que* $x_i = x_{i+1} = ... = x_k$*;*
- x_i *é um mínimo direito se* $x_i < x_{i+1}$ *e ainda* $\exists k < i$ *tal que* $x_k = x_{k+1} = ... = x_i$*;*
- x_i *é um mínimo plano se* $\exists k > i, l < i$ *tal que* $x_{l-1} > x_l = x_{l+1} = ... = x_i = ... = x_k < x_{k+1}$*.*

Definição 10 *Seja* $X = x_1, x_2, ..., x_n$ *uma série temporal. Seja também* x_i *um elemento desta série temporal, com* $1 < i < n$*. Então:*

- x_i *é um máximo estrito se* $x_{i-1} < x_i$ *e* $x_i > x_{i+1}$*;*
- x_i *é um máximo esquerdo se* $x_i > x_{i-1}$ *e ainda* $\exists k > i$ *tal que* $x_i = x_{i+1} = ... = x_k$*;*
- x_i *é um máximo direito se* $x_i > x_{i+1}$ *e ainda* $\exists k < i$ *tal que* $x_k = x_{k+1} = ... = x_i$*;*
- x_i *é um máximo plano se* $\exists k > i, l < i$ *tal que* $x_{l-1} < x_l = x_{l+1} = ... = x_i = ... = x_k > x_{k+1}$*.*

Na Figura 3.1 são mostrados vários extremos (círculos). Os pontos F, H, J, M e O são mínimos estritos. O ponto B é um mínimo esquerdo. O ponto C é um mínimo direito. O ponto P é um mínimo plano. Os pontos A, G, I, L e N são máximos estritos. O ponto D é um máximo esquerdo. O ponto E é um máximo direito. O ponto Q é um máximo plano.

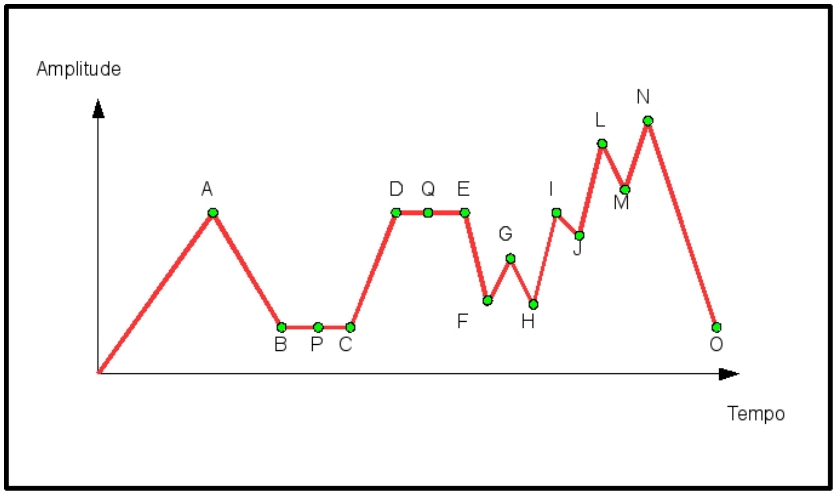


Figura 3.1: Alguns exemplos de extremos relativos

A partir do conceito de extremo, podem ser definidos os conceitos de importância de extremos [Gan03]. Além disso este trabalho define o conceito de prevalência de extremos e ainda o de extremo relativo.

Definição 11 *Seja $X = x_1, x_2, ..., x_n$ uma série temporal. Seja também x_i um elemento desta série temporal, com $1 < i < n$. Então x_i é um extremo com importância K e prevalência P se existe um intervalo $[il, ir]$, no qual x_i é extremo, tal que $il < i < ir$ e ainda:*

- $|i - il|, |i - ir| \geq P$;
- $dist(x_i, x_{il}) \geq K$ *e ainda* $dist(x_i, x_{ir}) \geq K$.

Definição 12 *Um extremo com importância K e prevalência P é denominado extremo relativo.*

A definição 12 pode ser entendida como uma forma de flexibilizar o conceito de extremos. No mundo real, qualquer sinal ou amostra é obviamente contaminada por ruídos. Assim, uma forma de lidar com estes ruídos na análise de uma série temporal é estabelecer métricas para considerá-los ou não; a prevalência P e a importância K de um extremo relativo é uma forma de lidar com ruídos ou ainda, estabelecer níveis para análise de uma série temporal. A Figura 3.2 mostra máximos e mínimos relativos, com prevalência 5. Além disso, os conceitos de importância e prevalência são exemplificados de forma gráfica. Na figura em questão, são selecionados todos os extremos que tenham pelo menos prevalência 5, isto é são extremos de todos os pontos anteriores e posteriores em uma distância de 5. Já o conceito de importância visa selecionar extremos que atingem uma amplitude mínima na região onde são considerados extremos.

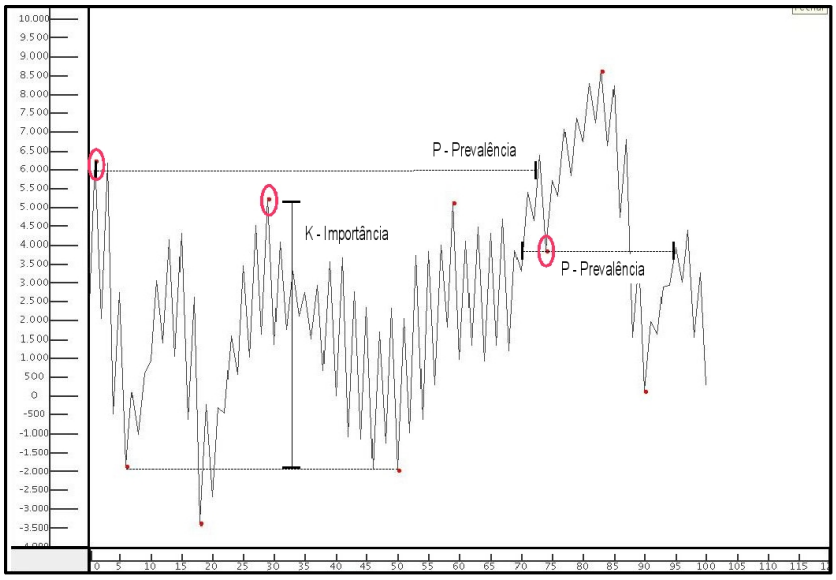


Figura 3.2: Máximos e mínimos relativos com prevalência 5

3.2 Extremos Relativos com Importância e Prevalência

Para facilitar o entendimento das definições seguintes, esta seção introduzirá alguma terminologia básica. A primeira terminologia é a representação

baseada em extremos relativos, que terá como parâmetros uma importância mínima K e uma prevalência mínima P; em seguida, outros conceitos são delineados, como o conceito de monte.

Definição 13 *Seja $X = x_1, x_2, ..., x_n$ uma série temporal. Sejam também as restrições de prevalência P e importância K; então :*

- $Max_{P,K}(X) = \{x_i \in X | x_i$ *é máximo estrito, com prevalência P e importância $K\}$.*
- $EMax_{P,K}(X) = \{x_i \in X | x_i$ *é máximo esquerdo, com prevalência P e importância $K\}$.*
- $DMax_{P,K}(X) = \{x_i \in X | x_i$ *é máximo direito, com prevalência P e importância $K\}$.*
- $PMax_{P,K}(X) = \{x_i \in X | x_i$ *é máximo plano, com prevalência P e importância $K\}$.*
- $Min_{P,K}(X) = \{x_i \in X | x_i$ *é mínimo estrito, com prevalência P e importância $K\}$.*
- $EMin_{P,K}(X) = \{x_i \in X | x_i$ *é mínimo esquerdo, com prevalência P e importância $K\}$.*
- $DMin_{P,K}(X) = \{x_i \in X | x_i$ *é mínimo direito, com prevalência P e importância $K\}$.*
- $PMin_{P,K}(X) = \{x_i \in X | x_i$ *é mínimo plano, com prevalência P e importância $K\}$.*

3.3 Representação Baseada em Extremos Relativos

A partir dos conceitos de $Max_{P,K}$, $EMax_{P,K}$, $DMax_{P,K}$, $PMax_{P,K}$, $Min_{P,K}$, $EMin_{P,K}$, $DMin_{P,K}$, $PMin_{P,K}$, um sinal pode ser representado, com perdas, de uma forma alternativa, tendo a prevalência P e a importância K fixadas. Assim tem-se que a RBER, de um sinal X, pode ser assim definida:

Definição 14 *Seja $X = x_1, x_2, ..., x_n$ uma série temporal. Sejam também as restrições de prevalência P e importância K; então a representação baseada em extremos relativos - $RBER_{P,K}$ - pode ser definida como a união de $Max_{P,K}(X)$, $EMax_{P,K}(X)$, $DMax_{P,K}(X)$, $Min_{P,K}(X)$, $EMin_{P,K}(X)$, $DMin_{P,K}(X)$.*

É interessante observar que $PMax_{P,K}(X)$ e $PMin_{P,K}(X)$ foram excluídos da definição 14 propositadamente, uma vez que os mesmos representam uma redundância desnecessária na representação. Pode-se ainda escrever, sem perda de generalidade:

Definição 15 *Dado um sinal X, $RBER_{P,K} = \{x_{e_1}, x_{e_2}, ..., x_{e_n}\}$ é o conjunto de extremos relativos do sinal X, com prevalência P e importância K, excluindo-se os máximos planos e os mínimos planos. Além disso, $1 < e_1 \leq e_n \leq |X|$.*

Para efeito comparativo a Figura 3.3 mostra como seriam as diversas representações nas prevalências 20,10,5 e 2.

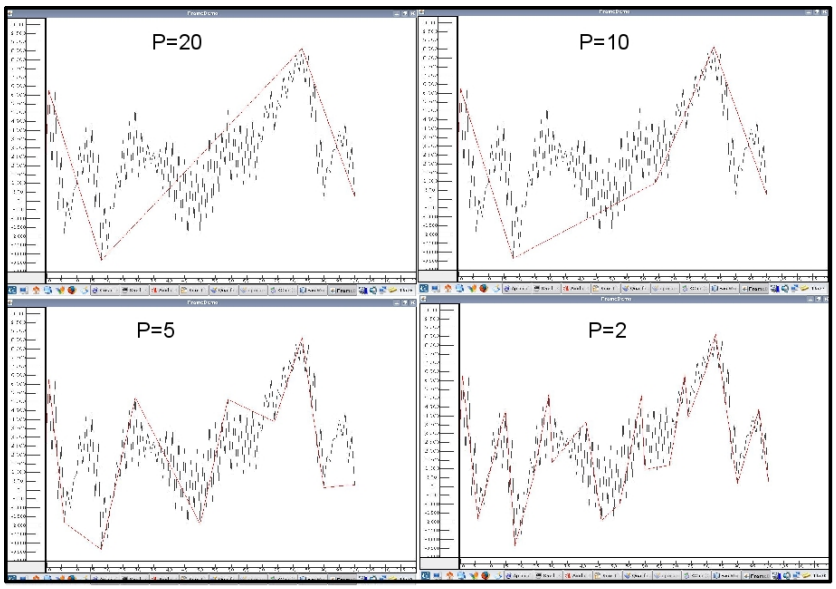


Figura 3.3: Comparação de representações em diversas prevalências

Os algoritmos 9, 10, 11,12 convertem um sinal discreto para a RBER, sem considerar a importância K, de forma a simplicar a análise através da diminuição de parâmetros.

Algorithm 9 Inicializa_Estruturas: Inicialização de variáveis para conversão RBER

Entrada: Prevalência :P

Saída: As variáveis $maximos$,$minimos$,$ppicos$,$pbordas$,$nfila$,Aux,$npicos$ são corretamente inicializadas.

Método: As variáveis são inicializadas da seguinte maneira:

$maximo \leftarrow 1$
$minimo \leftarrow 1$
$Appico \leftarrow P + 1$\{limite dos extremos\}
$Apborda \leftarrow 2P + 1$\{limite para se desconsiderar um extremo\}
$nfila \leftarrow 2P$\{Tamanho máximo do vetor auxiliar\}
$Aux \leftarrow \{0, 0, 0, ..., 0, 0\}$ \{vetor circular de $nfila$ posições\}
$npicos \leftarrow 0$

Algorithm 10 Acha_Mínimo (v,n,i,j) : Acha o mínimo de um vetor circular v, entre os índices i e j

Entrada: Um vetor circular v de tamanho n, e os índices i e j

Saída: Um índice x é encontrado tal que $v[i], v[j] \geq v[x]$

Método: O algoritmo funciona assim:

$x \leftarrow i$
for $k = i + 1, ..., j$ **do**
  if $v[k \bmod n] < v[x \bmod n]$ **then**
    $x \leftarrow k$
  end if
end for
return x

Algorithm 11 Acha_Máximo (v,n,i,j) : Acha o máximo de um vetor circular v, entre os índices i e j

Entrada: Um vetor circular v de tamanho n, e os índices i e j

Saída: Um índice x é encontrado tal que $v[i], v[j] \leq v[x]$

Método: O algoritmo funciona assim:

$x \leftarrow i$
for $k = i + 1, ..., j$ **do**
  if $v[k \bmod n] > v[x \bmod n]$ **then**
    $x \leftarrow k$
  end if
end for
return x

Algorithm 12 Conversão de um sinal $x[n] = \{x_1, x_2, ..., x_n\}$ para a representação $RBER_{P,0}$

Entrada: Sinal: $X = x_1, x_2, ..., x_n$; Prevalência: P
Saída: Representações: $RRBER_{P,0}$.
Método: As representações são preenchidas da seguinte maneira:

```
Inicializa_Estruturas ()
for i = 1, ..., n do
  ppico ← i − Appico
  pborda ← i − Apborda
  y ← X[i]
  Aux[i mod nfila] ← y
  if y > Aux[maximo mod nfila] then
    maximo ← i
  end if
  if y < Aux[minimo mod nfila] then
    minimo ← i
  end if
  if maximo = ppico then
    Adicionar ponto (i, X[i]) a R
  else
    if maximo = pborda then
      maximo ← Acha_Máximo(Aux, nfila, pborda + 1, i)
    end if
  end if
  if minimo = ppico then
    Adicionar ponto (i, X[i]) a R
  else
    if minimo = pborda then
      minimo ← Acha_Mínimo(Aux, nfila, pborda + 1, i)
    end if
  end if
end for
Retorne R
```

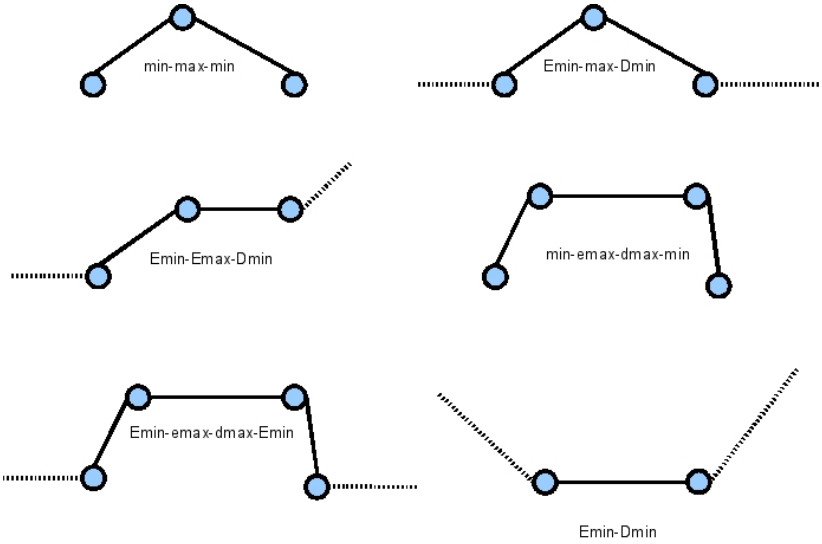


Figura 3.4: Seqüências de extremos relativos representando montes

3.4 Montes

A representação RBER é útil para analisar uma série temporal. Um conceito importante neste trabalho, que pode ser utilizado para aprimorar a representação RBER é o conceito de montes. A definição 16 apresentada o conceito de monte.

Definição 16 *Seja um sinal X, e sua representação $RBER_{P,K} = \{x_{e_1}, x_{e_2}, ..., x_{e_n}\}$, com prevalência P e importância K. Um monte M é uma seqüência de extremos relativos em $RBER_{P,K}$ que começa e termina em mínimos relativos, sem mínimos relativos intermediários.*

A partir da definição 16, pode-se dizer que as seguintes seqüências enquadram-se no conceito de monte:

- Min,Max,Min;
- $EMin$,Max,$DMin$;
- $EMin$,$EMax$,$DMin$;

- Min,$EMax$,$DMax$,Min;
- $EMin$,$EMax$,$DMax$,$DMin$;
- $EMin$,$DMin$;

As seqüências em questão podem ser representadas graficamente através da Figura 3.4.

3.5 Quantização Vetorial de Montes

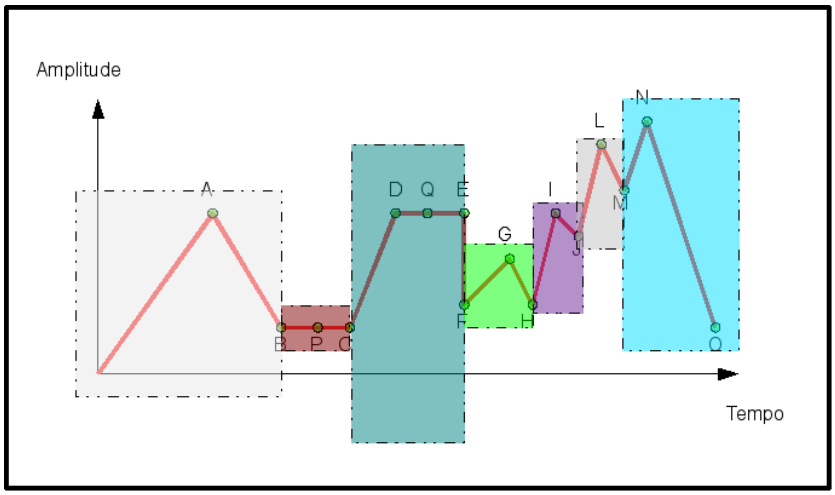


Figura 3.5: Marcação de Montes em um sinal

Na análise de problemas diversos, é útil eliminar a redundância e diminuir a complexidade da análise. No contexto deste trabalho, tal complexidade é diminuída através da quantização ou clusterização de montes similares. A idéia é analisar os montes similares de um determinado sinal X de maneira unificada. Isto reduz a complexidade da etapa de treinamento e reconhecimento.

Para que a quantização de montes possa ser empregada em um sinal X é necessário, que antes de mais nada, os montes possam ser identificados no sinal, conforme a regra da definição 16. A Figura 3.5 ilustra os montes marcados sob a Figura 3.1.

Após a marcação de montes, em um sinal digital, algoritmos de quantização vetorial podem ser empregados. Existem diversas formas de realizar a quantização vetorial. Os algoritmos possíveis são o k-médias, k-médias++[AV07], c-médias, k-medianas. A variedade é imensa. Neste trabalho, utilizou-se o k-medianas, por suas características únicas, entre elas a de não utilizar a operação de média para encontrar o centróide que representa o cluster.

Um aspecto importante relacionado à quantização vetorial é a escolha da função de similaridade para comparação de montes. Tal escolha pode influenciar no desempenho do classificador que usa a TER.

3.6 Representação Baseada em Extremos Relativos Quantizada (RBERQ)

Após a quantização vetorial de montes, uma outra representação do sinal estudado pode ser gerada. Para cada monte marcado no sinal é relacionado o índice do cluster, onde o monte foi quantizado. Em outras palavras, o processo de quantização vetorial gera um *codebook* que pode ser utilizado para representar o sinal de outra maneira. Ao invés de uma lista de números, o sinal pode ser representado como uma lista de símbolos, uma *codeword* para cada monte.

A representação de um sinal na forma de símbolos provenientes do procedimento de quantização de montes é denominada Representação Baseada em Extremos Relativos Quantizada (RBERQ).

Definição 17 *Seja S um conjunto de sinais $\{X_1, X_2, ..., X_{|S|}\}$, com suas respectivas representações $RBER(X_i)_{P,K}$. Após um procedimento de marcação de montes, seja $\mathrm{M} = \{m_1, ..., m_{|\mathrm{M}|}\}$ a lista de montes de todos os sinais em S. Seja também M_i o conjunto de montes de X_i, com $1 \leq i \leq |S|$. Após a execução de qualquer algoritmo de quantização vetorial, sobre o conjunto M, considerando uma função de distância qualquer, é produzido um conjunto de clusters $C = \{c_1, c_2, ..., c_{|C|}\}$. Os sinais de S podem ser representados de maneira alternativa, tendo como alfabeto o conjunto de clusters C. Define-se como $RBERQ(X_i)_{M,z} = \{s_1, s_2, ...s_{|M_i|}\}$ a representação produzida a partir de $RBER(X_i)_{P,K}$ e M, sendo que $z = |C|$. Além disso, $s_i \in C$ para qualquer*

elemento de $RBERQ(X_i)_{M,z}$*, obtido verificando-se qual é o cluster adequado para cada elemento de* M_i*.*

A partir do momento que sinais estudados são codificados como uma seqüência de símbolos, os mesmos sinais podem ser comparados através de funções de distância para cadeias. Assim, Classificadores, Algoritmos e Técnicas desenvolvidos para trabalhar com cadeias podem ser utilizados.

3.7 Quantização Baseada em Extremos Relativos (QBER)

A técnica QBER é uma metodologia para conversão de sinais digitais em símbolos, aqui denominada conversão simbólica de sinais digitais. Esta metodologia tem diversas etapas, conforme demonstra a Figura 3.6.

- Conversão dos Sinais Digitais para RBER: Os sinais são convertidos para RBER através do algoritmo 12, onde não se considera a importância K;
- Marcação de Montes: Todos os montes que começam e terminam em mínimos relativos, sem mínimos intermediários, viram montes, conforme definição 16, em todos os sinais representados via RBER.
- Normalização: Varia conforme o problema e a aplicação. Na implementação de referência, removeu-se o efeito da posição do monte (translação) e escala na amplitude, através da equação 1b:

$$MinM = Min_{i=0}^{|M|}\{(\frac{M(i)}{M(0)} - 1)\} \tag{1a}$$

$$MN(i) = (\frac{M(i)}{M(0)} - 1) - MinM \tag{1b}$$

- Clusterização de Montes: Cada monte normalizado compõe uma lista de montes que será clusterizada via qualquer algoritmo de clusterização. Neste texto utilizou-se a clusterização via k-medianas com o algoritmo PAM-SLIM [BRTJ06]. Este passo gera uma lista de centróides de tamanho k, onde k é um parâmetro para o algoritmo k-medianas. Um

requisito essencial deste passo é a escolha da função de distância para sinais utilizada;

- Tradução: Cada monte, em cada sinal, é comparado com a lista de centróides gerada no passo anterior. Associando-se cada centróide a um símbolo, cada sinal pode ter seus montes traduzidos para o símbolo associado à centróide mais similar. Em outras palavras, os sinais digitais são convertidos em cadeias ou, para RBERQ.

Após todo processo, o sinal é transformado em uma seqüência de símbolos, que pode ser utilizada por classificadores que lidam com símbolos.

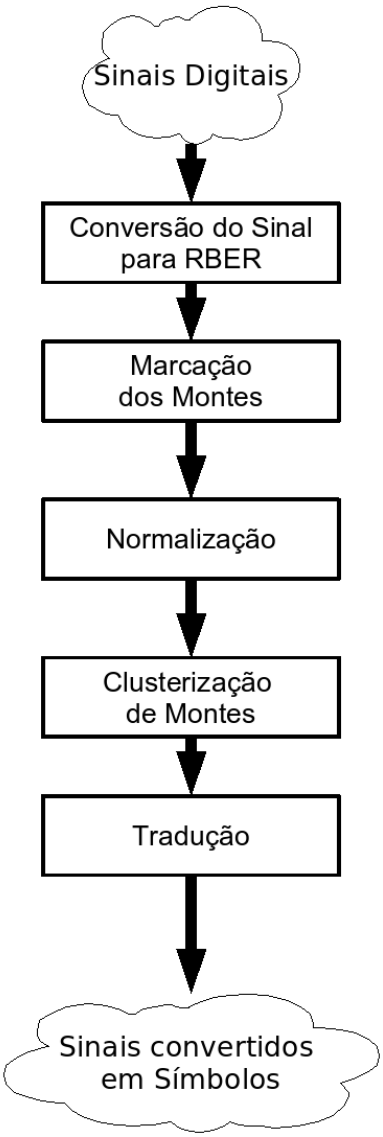


Figura 3.6: Etapas da Quantização Baseada em Extremos Relativos (QBER)

Capítulo 4

Implementação e Resultados

O principal objetivo desta pesquisa foi a formalização da técnica QBER. Para que se pudesse avaliar a utilidade de tal ténica, foi necessário o desenvolvimento de uma aplicação que a utilizasse. A aplicação efetua duas tarefas básicas: A conversão de sinais digitais em símbolos, através da TER e RBERQ e, a classificação destes sinais através de um classificador kNN simples.

A conversão simbólica utiliza a métrica ERP (Seção 2.1.3), já definida anteriormente. Já o classificador kNN compara símbolos dos objetos de treinamento com os símbolos dos objetos avaliados através da função de distância de Levenshtein, uma outra métrica, definida a partir da equação 4 (Seção 2.1.2).

4.1 Ferramentas Utilizadas

Diversas ferramentas e tecnologias foram utilizadas no desenvolvimento da implementação de referência:

1. Computador: Celeron M, 1.6 Ghz - com 1.2 Gb de memória RAM;
2. Sistema Operacional: Debian Linux;
3. Banco de Dados: Utilizou-se o banco de dados mysql;

4. Framework de persistência objeto-relacional : Hibernate;

5. Linguagens de programação: java e C++;

6. Bibliotecas: GBDI arboretum (ICMC/USP) (C++),boost (C++)

A idéia inicial era a codificação total em java. Porém, percebeu-se um grande ganho de performance ao utilizar a biblioteca GBDI arboretum, que implementa o MAM Slim-Tree. Daí optou-se por fazer um executável monolítico em C++ responsável pelo gerenciamento da Indexação métrica, e pela clusterização via PAM-Slim. O programa codificado em C++ possui uma interface simples, totalmente compatível com outros Sistemas Operacionais, bastando somente a recompilação.

As outras funções do programa, como armazenamento de informações e gráficos foram codificados em java, com vistas a incrementar a portabilidade. É interessante ressaltar que as otimizações e as tecnologias utilizadas foram partes relevante deste trabalho. Uma vez que sem elas não seria possível rodar uma quantidade tão grande de testes em tão pouco tempo. Os testes executados foram 280 variações de parâmetros. Sem otimização, cada variação demorava em média 65 minutos para executar (total de = 12 dias), isto já com a SlimTree sendo utilizada. Com as otimizações, cada variação passou a rodar em 2 minutos (total de = 9.3 horas).

4.2 Componentes do Classificador

O classificador proposto tem os seguintes componentes:

1. Uma lista de sinais discretos $A = \{a_1, a_2, ..., a_{|A|}\}$ - objetos de treinamento - representando classes de padrões. Assim, os objetos de treinamento apresentam rótulos especificando a qual classe pertencem.

2. Uma prevalência P a ser utilizada na representação RBERQ dos objetos de treinamento;

3. Uma representação simbólica associada a cada objeto de treinamento. Assim, cada amostra possui uma representação simbólica;

Os objetos de treinamento e a prevalência P são fornecidos como entrada. As representações simbólicas das amostras são obtidas na fase de conversão simbólica. Para cada objeto de treinamento existe uma representação simbólica, oriunda da QBER.

De posse de todos os componentes já descritos, é possível realizar o reconhecimento através do classificador kNN. Assim, tem-se como entrada a lista de objetos de treinamento juntamente com o objeto a ser avaliado, ambos em suas representações simbólicas. A saída é o percentual de reconhecimento de cada classe.

4.3 Fases do Modelo

O classificador proposto possui as fases de pré-processamento e reconhecimento. A Figura 1.1 mostra as fases de forma esquemática.

4.3.1 Fase de pré-processamento

Na fase de pré-processamento - detalhada na Figura 1.2 aplica-se a técnica QBER, com a finalidade de converter os sinais envolvidos em cadeias. Nesta fase são aplicados os procedimentos: Preparação dos Dados Iniciais; Coleta do Objeto a ser Avaliado; Conversão simbólica; Tradução simbólica.

Preparação dos Dados Iniciais

Inicialmente os objetos de treinamento - sinais digitais - são coletados e rotulados em uma classe específica. Este processo pode variar conforme a aplicação, portanto o classificador em si não precisa efetuar esta designação. O que ocorre é a importação dos objetos de treinamento via um arquivo XML padronizado[1]. Assim, o mesmo classificador pode ser usado para diferentes aplicações. Um programa externo pode gerar, se necessário, o arquivo XML para importação pelo classificador. Os objetos de treinamento, após a

[1]O XML é um formato padrão para representação de informações, facilmente codificado e utilizado em inúmeros sistemas e aplicações.

importação, são armazenados em um banco de dados do classificador, para posterior utilização. No exemplo apresentado neste trabalho - recomendações de compra de ações - um módulo específico da aplicação gera o arquivo XML dos objetos de treinamento. Outro detalhe importante é que podem ser escolhidos subconjuntos dos sinais cadastrados no banco de dados como objetos de treinamento.

Esta etapa ainda é útil para se aplicar algum filtro ou transformação sobre os objetos de treinamento. Tal transformação ou filtro não é obrigatória. Em nosso exemplo de aplicação - sugestões de compra para ações - inicialmente não se considerou a aplicação de filtros, também denominados indicadores na linguagem da análise técnica de ações. Porém, após concluídos os testes foram feitas simulações considerando a aplicação do indicador Índice de Força Relativa (*Relative strength index*) (RSI) [Jr.78], muito utilizado por especialistas em análise técnica de ações. A aplicação de uma transformação, filtro ou indicador sobre os objetos de treinamento pode fazer com que os padrões de reconhecimento fiquem mais evidentes e fáceis de se identificar.

Coleta do Objeto a ser Avaliado

Outro processo é a Coleta do Objeto a ser Avaliado. Pode ser simplesmente fornecido um arquivo contendo o objeto a ser classificado, em determinado formato, ou pode ser solicitado do classificador o reconhecimento de um objeto já cadastrado no banco de dados. Isto é útil para o processo de validação do classificador. É claro que em nosso exemplo de aplicação, como recomenda a literatura, o objeto a ser classificado não está entre os objetos de treinamento. Também quando se utiliza um filtro ou indicador sob os objetos de treinamento, a mesma transformação deve ser aplicada no objeto a ser avaliado - também um sinal de digital - de forma a uniformizar o domínio da análise.

Conversão simbólica

No procedimento de conversão simbólica, detalhado na Figura 1.2, todos os objetos de treinamento são convertidos em cadeias cujos símbolos pertencem a um alfabeto reduzido. Assim, a entrada é uma lista de sinais digitais unidimensionais $A = \{a_1, a_2, ..., a_{|A|}\}$. Para transformar os sinais digitais em

cadeias utiliza-se a técnica QBER (Seção 3.7). As etapas do processo são as seguintes:

1. Conversão do sinal para RBER: Conversão dos sinais $A = \{a_1, a_2, ..., a_{|A|}\}$ para a representação baseada em extremos relativos, com prevalência P;
2. Marcação dos montes: As combinações aceitáveis de montes como mínimo-máximo-mínimo, mínimo-Emáximo-Dmáximo-mínimo, mínimo-Emáximo-Dmáximo, Emáximo-Dmáximo-mínimo são marcadas, formando os "montes"dos objetos de treinamento.
3. Normalização: Cada "monte"gerado, em cada objeto de treinamento é normalizado. O procedimento de normalização varia conforme o problema atacado.
4. Clusterização de montes: Nesta etapa os "montes"da etapa anterior são quantizados. Para se realizar esta quantização pode ser utiliza-se o algoritmo PAM-SLIM, com a função de distância ERP.
5. Conversão: Cada monte, em cada sinal, é comparado com uma lista de centróides gerada no passo anterior. Associando-se cada centróide a um símbolo, cada sinal pode ter seus montes traduzidos para o símbolo associado à centróide mais similar.

O procedimento de conversão simbólica usa diversas estruturas e algoritmos para atender aos objetivos da quantização. Após a quantização dos "montes"cada cluster terá um representante único. Estes representantes são organizados em uma lista após a quantização. Isto é útil para a fase de tradução simbólica executada mais à frente.

Tradução simbólica

Já no procedimento tradução simbólica, os objetos de treinamento já estão convertidos para a representação simbólica. Agora, para que o reconhecimento possa ser executado é necessário a tradução do objeto a ser avaliado, onde serão reconhecidos padrões semelhantes aos objetos de treinamento.

Este processo é quase igual ao procedimento de conversão simbólica; Assim tem-se as etapas de conversão de representação e separação de "montes". A única etapa diferente é a etapa de "Quantização de montes", a qual é inexistente. A tradução de "montes"utiliza das estruturas criadas durante o processo de conversão simbólica, basicamente a Slim-Tree de centróides.

A tradução monte-a-monte do objeto a ser avaliado funciona assim: Lê-se cada monte um a um e realiza-se a consulta *k-nearest neighbors* com $k = 1$ para cada monte lido. O monte é "traduzido"para o símbolo que representa o cluster cujo representante é o mais semelhante ao monte lido.

4.3.2 Fase de reconhecimento

A fase de reconhecimento é a última fase - veja a Figura 1.3. Neste ponto já se têm os objetos de treinamento e o sinal alvo convertidos para a representação simbólica. Diante disto, basta utilizar as representações simbólicas como entrada para o classificador kNN, conforme descrito anteriormente.

Para otimizar a fase de reconhecimento, é gerada uma estrutura de indexação métrica dos objetos de treinamento já convertidos em símbolos. A diferença aqui é a métrica empregada e o tipo de objeto utilizado - cadeias. Assim, utiliza-se a distância de Levenshtein para comparar cadeias, construindo-se então uma Slim-Tree de cadeias. A otimização fica por conta da execução da consulta de vizinhos mais próximos, que agora é executa diretamente na Slim-Tree. Assim tem-se um ganho nítido de performance na execução do classificador kNN, uma vez que não se utiliza busca seqüencial.

A partir do instante que o classificador kNN está devidamente preparado e otimizado, podem ser fornecidos objetos para classificação. Um objeto a ser avaliado, fornecido ao classificador kNN têm como resposta os k objetos de treinamento mais similares (pela distância de Levenshtein), com seus respectivos rótulos de classe.

O último passo é interpretar a saída do classificador para gerar um resultado que possa ser utilizado. Em geral, o objeto a ser avaliado é reconhecido como pertencente à classe mais freqüente nos k objetos de treinamento mais similares. Porém, em nosso exemplo real, a interpretação da saída do classificado é outra. A forma de intepretar a saída do classificador kNN é exposta na seção 4.4.

4.4 Resultados

O objetivo deste capítulo é avaliar a utilidade da representação RBERQ através de um problema real. O sinal analisado foi uma série composta das cotações diárias, desde o ano de 1991, do ativo PETR4, uma ação negociada na Bolsa de Valores de São Paulo. O objetivo consistiu na obtenção de recomendações confiáveis de compra do ativo, de forma que o especulador pudesse obter um lucro mínimo de 2% por transação realizada na Bolsa de Valores.

O interesse é a confiabilidade do classificador em relação às recomendações de compra; tais recomendações devem ter uma boa confiabilidade, de forma que ao final tenha-se uma esperança matemática positiva no investimento. Tal problema contém uma componente aleatória grande, portanto o interesse não é prever todo comportamento do mercado e sim obter padrões seguros de compra.

O objetivo é tão somente avaliar a "sanidade"de todas as teorias propostas. Até porque o que foi proposto é uma representação, e a distância empregada nas representações simbólicas, neste trabalho, é a distância de Levenshtein. Muitas distâncias diferentes podem ser empregadas para descobrir diferentes tipos de padrões, porém já existe satisfação no emprego da distância de Levenshtein, uma vez que o objetivo da classificação foi atingido - indicações confiáveis de compra, com vistas a obter uma esperança matemática positiva para a simulação do investimento.

Outras questões econômicas estão envolvidas, como a Teoria da eficiência do mercado, porém não pretendemos adentrar nestas questões por se tratarem de discussões do campo da Ciência Econômica.

4.4.1 Preparação dos Dados

Os dados foram preparados baseando-se nas cotações diárias das ações preferências da empresa Petrobrás, de código PETR4. Foram extraídos sinais diários de informação, considerando a quantidade de dias anteriores, a qual designamos "barras anteriores"e, ainda, para efeito de designar o rótulo da classe, consideramos o lucro máximo e o prejuízo máximo em uma quantidade de dias posteriores, denominados "barras posteriores".

Assim, cada sinal na base de dados, contém um código relativo ao ativo (PETR4), à data, às barras posteriores e anteriores e ao código da importação. Este sinal contém as amplitudes (abertura, máxima, mínima e fechamento) dos dias anteriores, aqui denominadas barras anteriores.

As classes consideradas estão na tabela 4.1. Cada sinal é portanto associado a duas classes: a classe de prejuízo máximo; a classe de lucro máximo. O objetivo é encontrar padrões onde o lucro máximo é acima de 2%. Em outras palavras, queremos padrões que têm baixa probabilidade de estar nas classes $L_$,$L0$ e $L1$.

Código da Classe	Descrição
L_	Lucro Máximo Inexistente
L0	Lucro Máximo entre 0% e 1%
L1	Lucro Máximo entre 1% e 2%
L2	Lucro Máximo entre 2% e 3%
L3	Lucro Máximo entre 3% e 4%
L4	Lucro Máximo entre 4% e 5%
L5	Lucro Máximo entre 5% e 6%
L6	Lucro Máximo entre 6% e 7%
L7	Lucro Máximo entre 7% e 8%
L8	Lucro Máximo entre 8% e 9%
L9	Lucro Máximo entre 9% e 10%
L10	Lucro Máximo acima de 10%
P_	Prejuízo Máximo Inexistente
P0	Prejuízo Máximo entre 0% e 1%
P1	Prejuízo Máximo entre 1% e 2%
P2	Prejuízo Máximo entre 2% e 3%
P3	Prejuízo Máximo entre 3% e 4%
P4	Prejuízo Máximo entre 4% e 5%
P5	Prejuízo Máximo entre 5% e 6%
P6	Prejuízo Máximo entre 6% e 7%
P7	Prejuízo Máximo entre 7% e 8%
P8	Prejuízo Máximo entre 8% e 9%
P9	Prejuízo Máximo entre 9% e 10%
P10	Prejuízo Máximo acima de 10%

Tabela 4.1: Código das Classes

Considerando os parâmetros envolvidos, a preparação de dados pode ter

dois parâmetros de variação, o número de barras posteriores e o número de barras anteriores. Em nossa análise os dados foram preparados de duas maneiras distintas, conforme a tabela 4.2.

Código da Importação	Barras Posteriores	Barras Anteriores
3_14	3	14
3_20	3	20

Tabela 4.2: Preparação dos Dados

4.4.2 Preparação do Classificador

A preparação do classificador consiste em gerar os símbolos dos objetos de treinamento considerados, via RBERQ. Para tal, deve-se considerar como parâmetros os objetos de treinamento e a quantidade de centróides para clusterização.

Para facilitar a análise dos dados e classificador, foi ainda definido o conceito de Grupo de Treinamento, que é simplesmente uma lista previamente definida de objetos de treinamento. Assim, o classificador real têm como parâmetros de preparação o código do Grupo de Treinamento, a quantidade de centróides para quantização e, ainda, a prevalência P.

Os dados utilizados têm históricos de cotação desde 1991. Como os anos que foram simulados na classificação foram os anos 2002, 2003, 2004, 2005, 2006, 2007, 2008, se considerou grupos de treinamento que continham todas informações anteriores ao ano avaliado, por exemplo, o ano 2002 considerou dados de 1991 a 2001. O fato do grupo de treinamento não conter o ano que será avaliado é um princípio de validação de classificadores na disciplina Reconhecimento de Padrões.

4.4.3 Reconhecimento dos Dados

O reconhecimento de dados, através do classificador preparado, foi realizado dia a dia, considerando o classificador preparado com o grupo de treinamento dos anos anteriores. Um parâmetro importante para o reconhecimento é o

valor de k (vizinhos mais próximos no classificador kNN), para fins de nossa avaliação, consideramos o k como 5, 10, 15.

Em cada dia avaliado o classificador simplesmente informa quais os percentuais das classes reconhecidas. Para converter a resposta do classificador em uma sugestão de compra, considerou-se como uma sugestão de compra do classificador aquela onde o somatório percentual das classes $L2 - L10$ foi superior a um parâmetro que denominados confiança. Foram feitos testes com confiança 80% e 60%.

Em cada sugestão de compra, o procedimento de validação do classificador verificou a realidade, isto é, se a sugestão de compra foi efetiva. Os casos em que não foram sugeridos compra, não foram considerados. Se uma sugestão de compra foi correta ela foi contabilizada na variável "Compras Corretas". Caso uma sugestão de compra resultou em lucro máximo inferior a 2%, contabilizou-se a sugestão na variável "Falsas Compras". Assim, a performance do classificador é avaliada pelo percentual de compras corretas, dentre as sugestões de compra geradas, considerando-se o ano em análise.

4.4.4 Resultados Obtidos

Após variarmos os parâmetros de preparação de dados, preparação do Classificador, e reconhecimento de Dados, foram encontrados os seguintes percentuais de compras corretas, conforme demonstram os gráficos. Nas Figuras 4.1 e 4.2 foram testados classificadores com diversas prevalências, centróides e barras_anteriores, mas com confiança 80% e 60% respectivamente; Além disso, tais classificadores consideraram todos os dados anteriores ao ano de análise.

As Figuras 4.3 e 4.4 apresentam a análise do desempenho médio de classificação para as confianças de 80% e 60% respectivamente. Para o primeiro caso, foi obtido um classificador com 73% médio de acerto, nas sugestões de compra, com parâmetros (10;1;5;20) para K, prevalência, barras anteriores e centróides, considerando um Grupo de Treinamento igual a todos os anos anteriores. Já para a confiança de 60% foram obtidos diversos classificadores cujos parâmetros resultam em uma taxa de 67% de desempenho médio: (5;2;14;20), (15;3;14;20), (15;3;14;10), (5;2;14;50), (5;2;14;10), (15;3;10;20), (10;2;14;50), (10;3;14;10), (15;2;14;50).

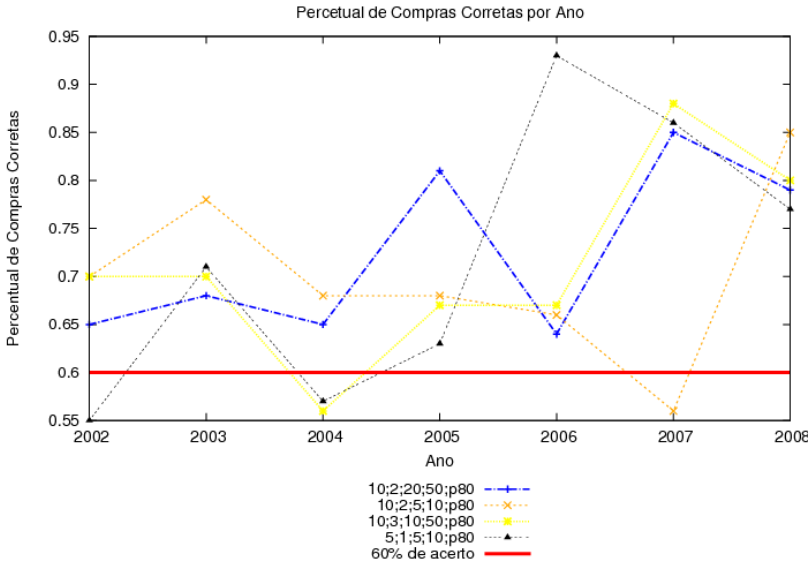


Figura 4.1: Percentual de compras corretas para $confiança = 80\%$, e preparação com todos os anos anteriores

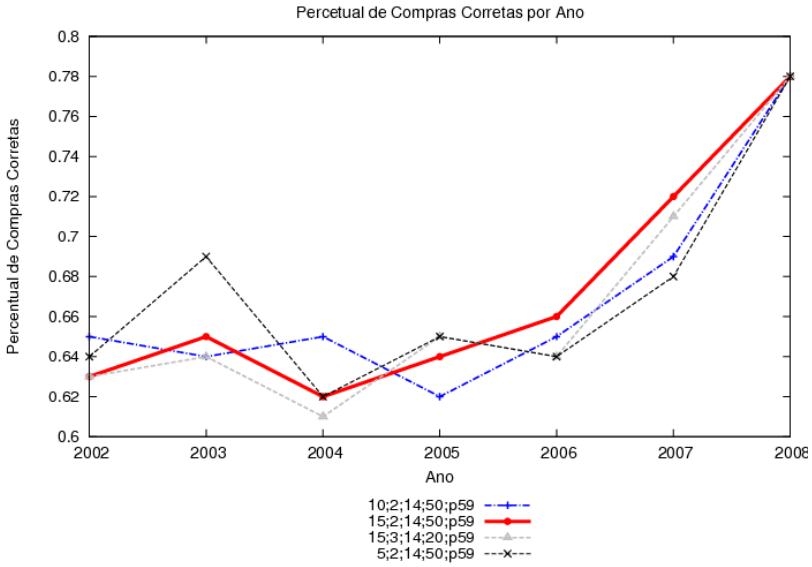


Figura 4.2: Percentual de compras corretas para $confiança = 60\%$, e preparação com todos os anos anteriores

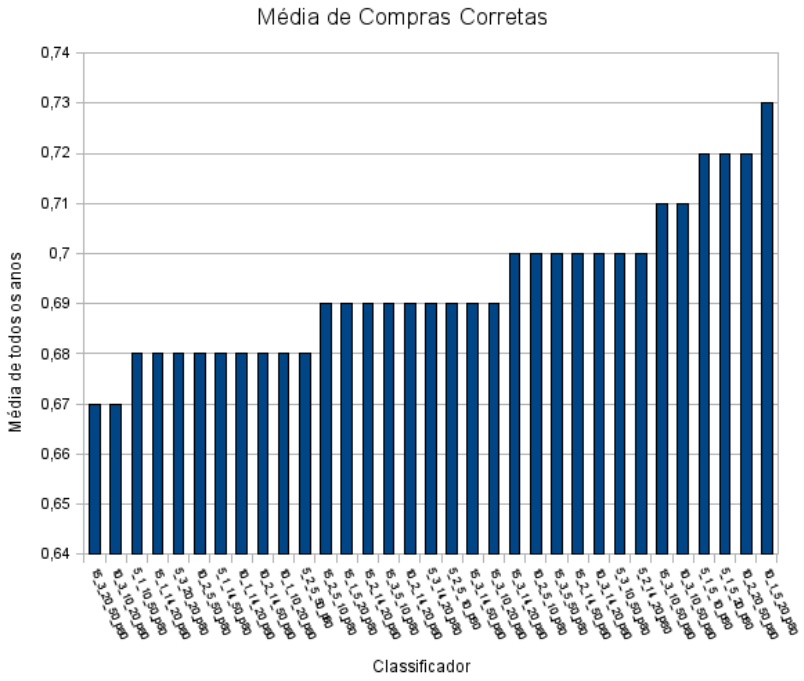


Figura 4.3: Média de compras corretas - $confiança = 80\%$, e preparação com todos os anos anteriores

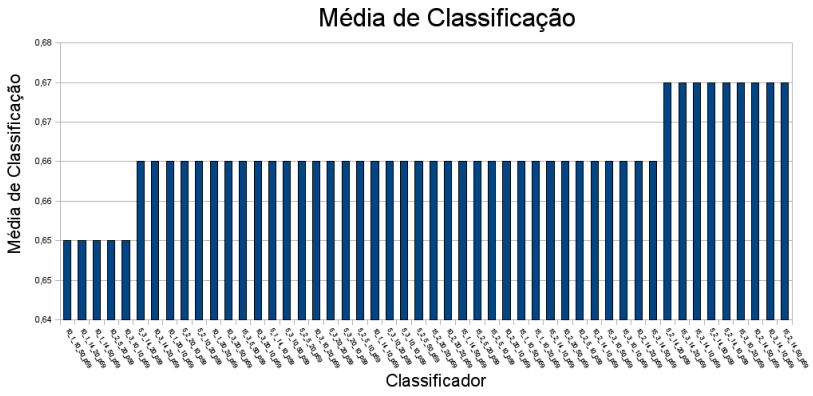


Figura 4.4: Média de compras corretas - $confiança = 60\%$, e preparação com todos os anos anteriores

4.4.5 Avaliação da Rentabilidade Anual

O que vimos até agora, resultante da simulação, foi a apresentação dos resultados do classificador, considerando o percentual de compras corretas, dentre as recomendações de compra geradas pelo classificador. A simulação da rentabilidade do investimento a partir dos indicativos do classificador é outra questão. Isto porque devem ser considerados outras variáveis; uma delas é a quantidade de recomendações gerada; a outra é a estratégia de investimento utilizada.

A estratégia de investimento visa uma rentabilidade média positiva, considerando a esperança matemática das operações. Um exemplo de estratégia é utilização de ordens de stop[2] no valor de 2% abaixo do capital investido por operação (stop de perda) e, ainda no valor de 2% acima do capital investido (stop de ganho), sem considerar as taxas de corretagem. Assim, em nosso caso, em uma análise simplista, ter-se-ia a seguinte esperança matemática por operação:

$$E_{oper} = 0,02 \times (Percentual_{Comprascorretas}) - 0,02 \times (1 - (Percentual_{Comprascorretas})) \quad (1)$$

Exemplificando, ao usar a fórmula da equação 1, teríamos, a grosso modo, ou lucro de 2% ou prejuízo de 2%. Por exemplo, considerando um percentual de compras corretas de 71%, teríamos uma rentabilidade média por operação de 0,84%. Agora o lucro anual depende da quantidade de operações realizadas, ou ainda a quantidade de sugestões de compras realizadas pelo classificador.

Portanto, o classificador mais rentável seria aquele que ao final do ano apresentasse um valor maior para :

$$Rentabilidade = E_{oper} \times Quantidade_{Oper} \quad (2)$$

Analisando sob o prisma de maximizar a rentabilidade anual, os classificadores (10;1;20;10;60%) e (5;1;20;10;80%) são mais adequados, conforme

[2]São ordens de venda ativadas automaticamente quando o ativo chega a um determinado valor

demonstram as Figuras 4.5 e 4.6.

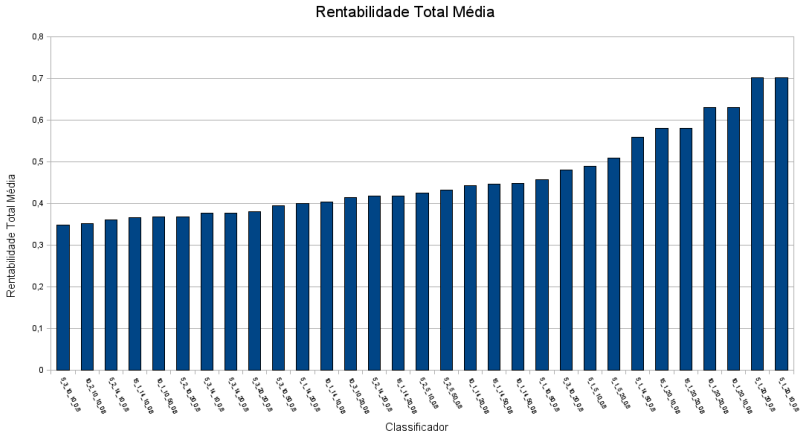


Figura 4.5: Rentabilidade anual média - $confiança = 80\%$

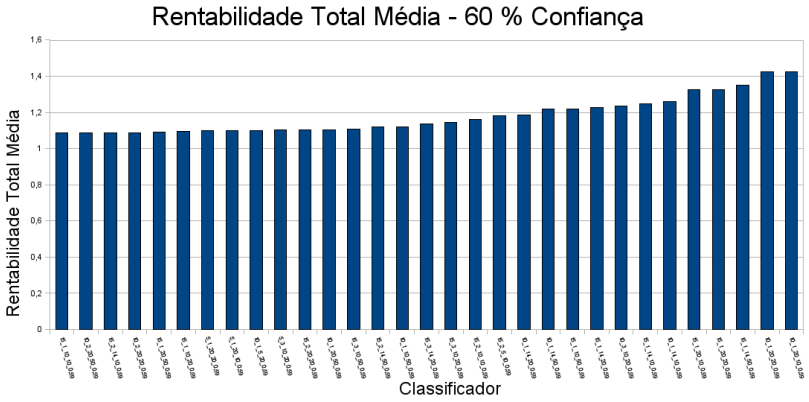


Figura 4.6: Rentabilidade anual média - $confiança = 60\%$

Outra questão a se considerar é a variabilidade no percentual de rendimento anual. Assim, sob esta ótica, não é adequado um classificador que possui um desvio padrão muito grande, em suas rentabilidades anuais. Assim, considerando-se um desvio padrão inferior a 35%, os classificadores (15;1;14;50;80%), (10;1;14;50;80%), (5;1;10;50;80%), (15;2;5;10;60%), (15;2;5;20;60%), (10;2;5;20;60%) são destacados na Figura 4.7.

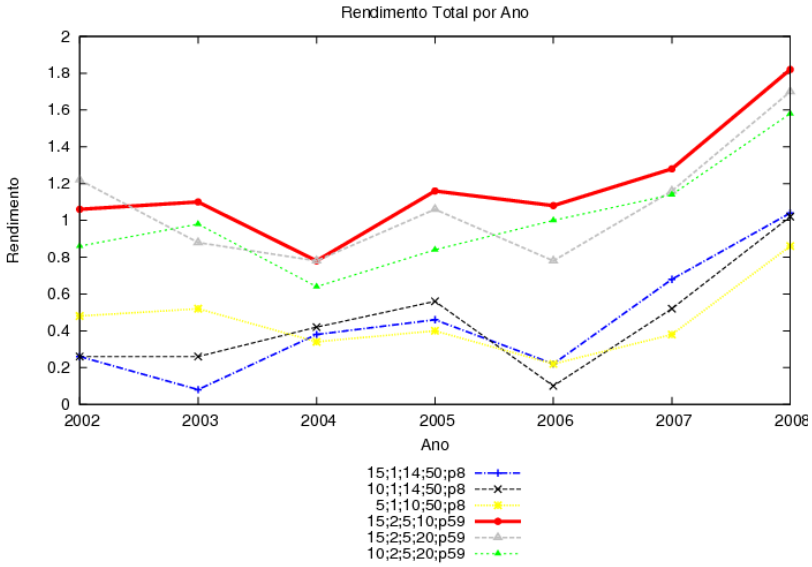


Figura 4.7: Rentabilidade anual - melhores classificadores

4.5 Avaliação da utilidade do método QBER, através da comparação de classificadores

Um aspecto importante que deve ser avaliado em qualquer método novo é quais são os ganhos obtidos pela utilização desse método. Aqui isto é obtido através da codificação de outro classificador kNN - aqui denominado kNN/ERP simples - que não usa o método QBER na fase de pré-processamento, mantendo-se todas as outras etapas semelhantes.

Como pode ser observado na Figura 4.8, os sinais são extraídos da mesma maneira, porém a fase de pré-processamento simplesmente indexa os sinais extraídos via SlimTree, usando a métrica para sinais ERP (Seções 2.1.3 e 2.2.2). Na fase de reconhecimento, utiliza-se a estrutura gerada para executar a busca por similaridade - kNN - mais eficientemente.

Os dois classificadores, kNN/ERP simples e o classificador original que usa QBER (kNN/QBER/Levenshtein), foram então comparados, considerando o parâmetro número de dias de informação, para cada sinal digital. Os parâmetros resultantes foram o tempo de preparação do classificador, o percentual de compras corretas e a rentabilidade total. O tempo de pre-

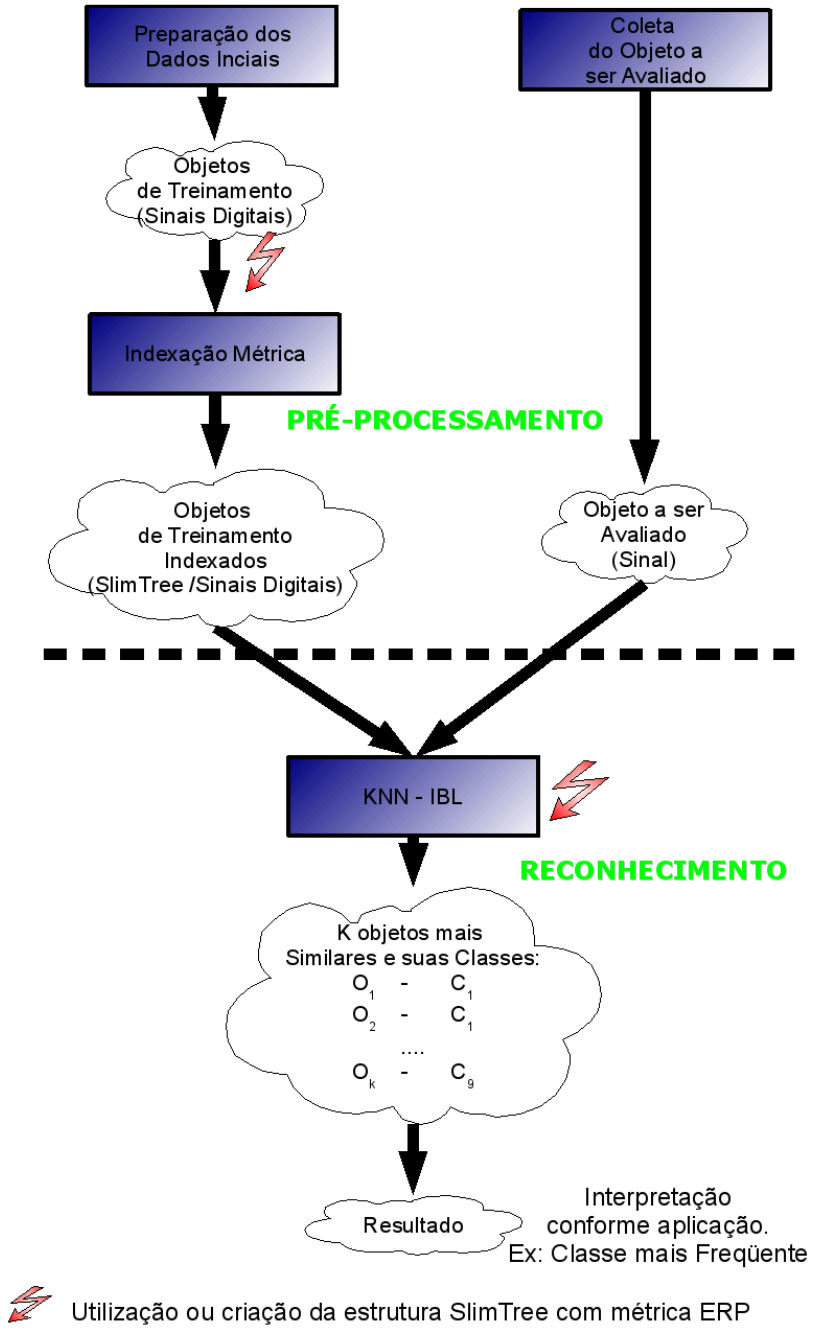


Figura 4.8: Arquitetura de um classificador Knn Simples, sem QBER

paração de um classificador e a performance do mesmo são variáveis são muitos relevantes para o reconhecimento de padrões em sinais digitais, pois conforme o tipo de problema atacado, pode não ser viável, por exemplo, esperar muito tempo para preparação do classificador e, além disso, os sinais podem ter comprimento razoável.

Assim, para comparação dos dois classificadores, um usando a QBER e outro sem usar a QBER, foram feitas diversas simulações, variando-se o parâmetro já citado. Foram testados sinais com 5,10,14 e 20 dias de informação; também foram considerados os parâmetros percentual de compras corretas, tempo de preparação e ainda a rentabilidade total das sugestões de compras, calculadas por classificador.

4.5.1 Tamanho do sinal versus Tempo de preparação do classificador

A primeira análise efetuada foi a avaliação do tempo de preparação do classificador em relação ao tamanho do sinal. A Figura 4.9 mostra o tempo de preparação de três classificadores - QBER1,QBER2 e ERP1 - em relação ao tamanho do sinal, em dias. Os classificadores apresentados são os que apresentaram os melhores desempenhos de classificação, considerando-se somente a utilização da métrica ERP (ERP1) e, considerando-se a utilização do método QBER (QBER1 e QBER2). Os parâmetros entre parênteses significam:

- ERP1 (N;3;15;Linear) : N - não utilização do método QBER, 3 - barras posteriores, 15 - sinais semelhantes considerados no kNN, Linear - Normalizador do tipo Linear;
- QBER1 (S;3;10;1;15;Null;14-rsi) : S - utilização do método QBER, 3 - barras posteriores, 10 - número de centróides, Null - Sem normalização, 14-rsi - Filtro RSI de 14 períodos;
- QBER2 (S;3;20;3;5;Linear) : S - utilização do método QBER, 3 - barras posteriores, 20 - número de centróides, Linear - Normalizador do tipo Linear.

Como pode ser observado na Figura 4.9, o classificador que não utiliza QBER apresentou um tempo maior em relação ao Tamanho do Sinal. Já os

classificadores que utilizam o método desenvolvido apresentam um tempo de preparação Linear.

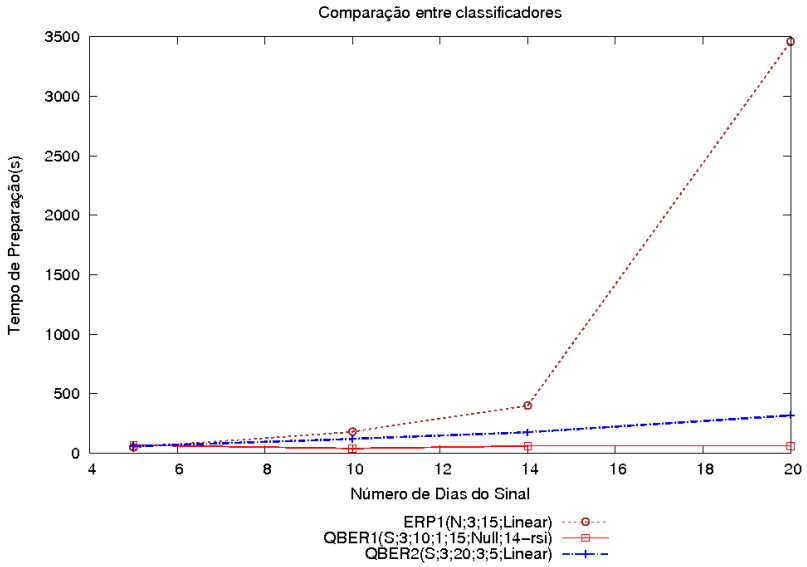


Figura 4.9: Tempo de Preparação versus Tamanho do Sinal

4.5.2 Tamanho do sinal versus Performance do classificador

A Figura 4.9 é um ponto positivo para justificar a teoria desenvolvida. Porém outros aspectos são importantes. O primeiro deles é a performance do classificador. Considerando nosso exemplo de aplicação, será utilizado o percentual de compras corretas, em função do tamanho do sinal analisado. A Figura 4.10 mostra os desempenhos dos 3 melhores classificadores, considerando-se o ano de 2008.

Como pode ser observado, para sinais pequenos, a simples aplicação da métrica supera o desempenho do classificador que utiliza QBER. Porém, ao se aumentar o tamanho do sinal em análise percebe-se que em alguns momentos a utilização do QBER é melhor, dependendo da combinação de parâmetros.

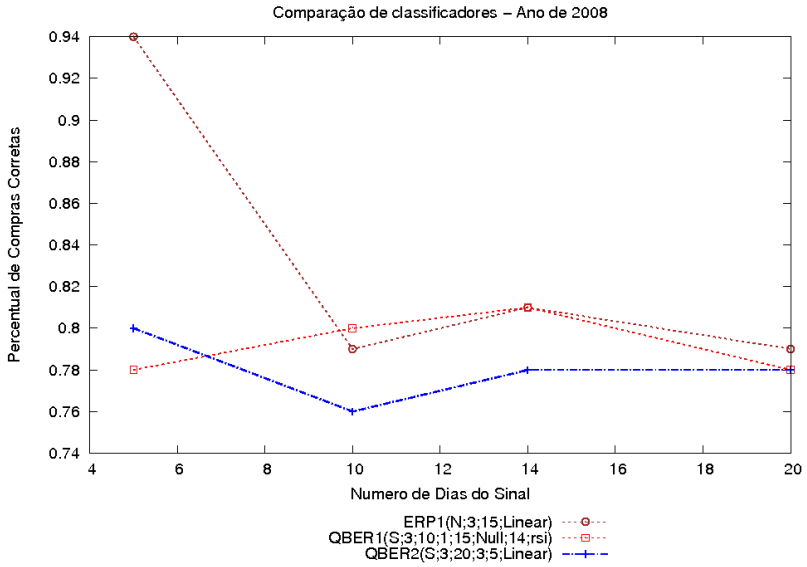


Figura 4.10: Percentual de Compras Corretas versus Tamanho do Sinal

4.5.3 Tamanho do sinal versus Rendimento Total

Finalizando a comparação entre os classificadores, é fundamental e decisiva a análise do parâmetro Rendimento Total. Conforme descrito na Seção 4.4.5, a rentabilidade total compreende a multiplicação da quantidade de sugestões de compra pela esperança matemática das operações. A Figura 4.11 apresenta os resultados dos 3 melhores classificadores identificados.

Aqui se tem a comprovação que a melhor escolha é a utilização da QBER na fase de pré-processamento, para o problema atacado. O classificador QBER1 tem um desempenho médio que supera, em 200%, o classificador ERP1, que não usa QBER.

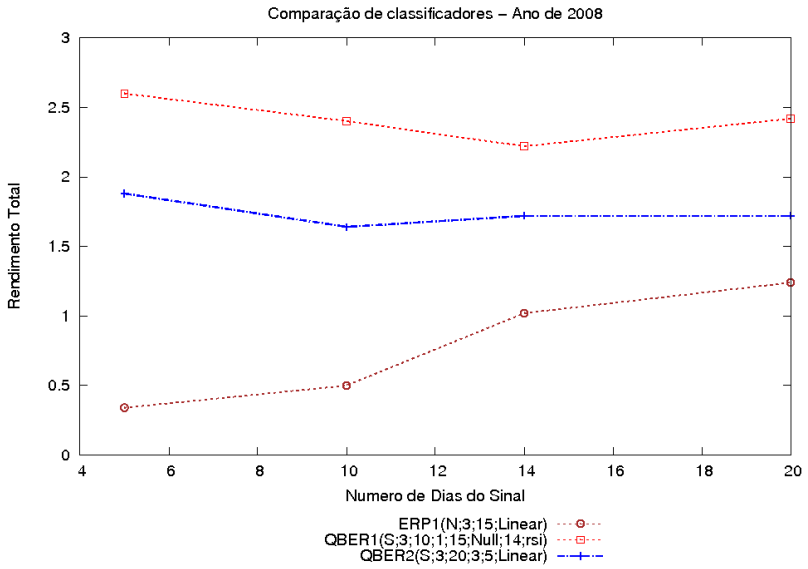


Figura 4.11: Rendimento Total versus Tamanho do Sinal

Capítulo 5

Conclusão

O estudo de técnicas para a classificação de padrões, em sinais digitais, é algo relevante. Existem variados tipos de informações que podem ser convertidas em sinais digitais; Informações como Eletrocardiogramas, Terremotos, Séries Financeiras, Histogramas de Imagens, Sinais de Áudio, são exemplos valiosos. Ao longo dos anos, diferentes representações e métodos de conversão têm sido propostos. Entre os métodos mais utilizados encontram-se as transformações de Fourier, as Wavelets, entre tantos métodos e alternativas. Cada representação gera todo um campo de pesquisa, com técnicas próprias e peculiares. Assim, a técnica QBER gera uma nova linha de pesquisa e com ela, oportunidades.

O objetivo principal deste trabalho foi, portanto, a formalização, avaliação e implementação da técnica QBER, que é uma técnica para conversão de sinais digitais em símbolos. As representações simbólicas são bem antigas e exaustivamente utilizadas em diversos algoritmos, metodologias e sistemas. Existem centenas de estruturas e modelos para a análise e indexação de seqüências de símbolos, textos etc. Eis o motivo pelo qual este trabalho se empenhou em viabilizar as técnicas aqui desenvolvidas.

É importante notar que tal técnica só se tornou viável devido desenvolvimentos notáveis em outros campos de pesquisa, como o desenvolvimento de estruturas para indexação métrica e o aprimoramento de algoritmos para clusterização. A partir destas inovações ficou computacionalmente viável a criação de uma técnica de conversão simbólica que tivesse como pré-suposto um processo de clusterização que usasse o algoritmo k-medianas. Nossa

pesquisa utiliza-se extensivamente da estrutura de indexação métrica Slim-Tree [TTSF00] e, além disso do algoritmo de clusterização derivado PAM-SLIM[BRTJ06].

Os benefícios da utilização de símbolos para análise de sinais digitais são enormes. Podem ser utilizados técnicas oriundas da Teoria da Computação, Biologia Computacional e áreas afins. Podem ser utilizadas funções de distância para cadeias, algoritmos de inferência gramatical, entre outros tantos.

É claro que muitas lacunas ainda existem para serem preenchidas. Como definir qual a função de distância adequada para um determinado problema? Qual o valor ideal para o parâmetro de prevalência no processo de conversão simbólica ? Como achar este valor rapidamente?

Este trabalho se limitou a formalizar o processo de conversão simbólica, bem como provar a utilidade e "sanidade"de tal metodologia. A implementação de referência poderia ser otimizada adequadamente. Com pouca dedicação à otimização do processo, se conseguiu otimizar um procedimento que demorava 60 minutos para execução em 2 minutos, em uma mesma máquina.

O caso de teste - Recomendações automáticas de compra de ações - teve um desempenho simulado satisfatório, atingindo rentabilidades anuais médias simuladas em torno de 70%, desconsiderando corretagens e outras questões. Para o "*bear market*[1]"de 2008, o classificador obteve rentabilidades superiores a 100%. É claro que existe a teoria de eficiência do mercado e caso algum dia este método se torne popular, as rentabilidades anuais tendem a cair.

Para chegarmos a estes resultados foram testados diversos parâmetros para o classificador desenvolvido, considerando os anos de 2002, 2003, 2004, 2005, 2006, 2007 e 2008. Os testes duraram cerca de 9h e testaram cerca de 280 variações de parâmetros; tais testes podem ser reproduzidos caso haja necessidade. O objetivo de provar a utilidade da representação desenvolvida foi plenamente alcançado.

[1]Ano em que a desvalorização da bolsa supera 20%.

5.1 Limitações

Cada problema pressupõe a adoção de procedimentos específicos. Não é diferente no método proposto. A preparação dos dados, ou ainda, dos objetos de treinamento inexoravelmente variará de aplicação para aplicação. O processo de normalização dos "montes"pode também variar. Em decorrência do tempo reduzido, o autor deste trabalho firmou o objetivo de provar a utilidade do método, assim, o caso de teste ficou reduzido somente a uma aplicação.

Também não se focou no estudo da econometria por trás dos resultados obtidos. Isto seria algo muito empolgante. Não foram aplicadas outras funções de distância para cadeias. Isto poderia revelar outros padrões e a quantidade de sugestões de compra e/ou a taxa de acerto poderia melhorar.

5.2 Aperfeiçoamentos Futuros

Muitas questões ficaram sem resposta. Tais questões poderiam ser objeto de trabalhos futuros:

1. Qual o valor ideal de prevalência?
2. Qual a quantidade mínima de objetos de treinamento ?
3. Qual a função de distância ideal para cadeias, no processo de análise dos símbolos gerados?
4. Qual a função de distância ideal para a quantização baseada em extremos relativos?
5. Com relação ao estudo de caso, como as simulações se comportariam em ativos diferentes?
6. Com relação ao estudo de caso, como as simulações se comportariam em operações *day trade*?
7. Como o modelo se comportaria na análise de outros tipos de sinais digitais?

8. Poderíamos realizar busca aproximada de padrões em sinais digitais ?
9. Qual o impacto da utilização do algoritmo aproximado PAM-SLIM sobre o reconhecimento ?
10. Como otimizar o modelo no caso de funções de distância não métricas ou ainda, pseudo-métricas ?

Em algumas destas sugestões, o autor teria o prazer de continuar a pesquisa.

Referências Bibliográficas

[AKA91] David W. Aha, Dennis Kibler, e Marc K. Albert. Instance-based learning algorithms. *Mach. Learn.*, 6(1):37–66, 1991. ISSN 0885-6125.

[App07] Apple. Core audio glossary, 2007.

[AV07] David Arthur e Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, páginas 1027–1035. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007. ISBN 978-0-898716-24-5.

[BRTJ06] Maria Camila N. Barioni, Humberto L. Razente, Agma J. M. Traina, e Caetano Traina Jr. An efficient approach to scale up k-medoid based algorithms in large databases. In *XXI Simpósio Brasileiro de Banco de Dados*, páginas 265–279. SBBD' 2006, São Carlos-SP, 2006.

[CN04] L. Chen e R. Ng. The marriage of lp-norms and edit distance, 2004.

[COO05] Lei Chen, M. Tamer Özsu, e Vincent Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, páginas 491–502. ACM Press, New York, NY, USA, 2005. ISBN 1-59593-060-4.

[CPRZ97] Paolo Ciaccia, Marco Patella, Fausto Rabitti, e Pavel Zezula. Indexing metric spaces with m-tree. In *Sistemi Evolui per Basi di Dati*, páginas 67–86. 1997.

[Fuc97] M. Fuchs. Instance-based learning by searching. *Intelligent Information Systems, 1997. IIS '97. Proceedings*, páginas 189–193, 8-10 1997.

[Gan03] H. Gandhi. Important extrema of time series: Theory and applications, 2003.

[Gun02] Dimitrios Gunopoulos. Discovering similar multidimensional trajectories. In *ICDE '02: Proceedings of the 18th International Conference on Data Engineering*, página 673. IEEE Computer Society, Washington, DC, USA, 2002.

[HU90] John E. Hopcroft e Jeffrey D. Ullman. *Introduction To Automata Theory, Languages, And Computation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990. ISBN 020102988X.

[IEE90] IEEE. Ieee standard glossary of image processing and pattern recognition terminology. *IEEE Std 610.4-1990*, páginas –, 26 Mar 1990.

[Jr.78] J.Welles Wilder Jr. *New Concepts In Technical Trading Systems*. Trend Research, 1978. ISBN 894590278.

[KSF+96] Flip Korn, Nikolaos Sidiropoulos, Christos Faloutsos, Eliot Siegel, e Zenon Protopapas. Fast nearest neighbor search in medical image databases. In *The VLDB Journal*, páginas 215–226. 1996.

[Kur01] Stefan Kurtz. Foundations of sequence analysis, July 2001.

[Lan68] Serge Lang. *CALCULO*, volume 1. ADDISON WESLEY, 1968.

[Lou02] Isabel M. G. Lourtie. *Sinais e Sistemas*. Escolar Editora, 2002. ISBN 972-592-130-5.

[MP07] Michael D. Morse e Jignesh M. Patel. An efficient and accurate method for evaluating time series similarity. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, páginas 569–580. ACM Press, New York, NY, USA, 2007. ISBN 978-1-59593-686-8.

[Nav01] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[NH94] Raymond T. Ng e Jiawei Han. Efficient and effective clustering methods for spatial data mining. In Jorge B. Bocca, Matthias Jarke, e Carlo Zaniolo, editores, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, páginas 144–155. Morgan Kaufmann, 1994. ISBN 1-55860-153-8.

[Pek05] Elżbieta Malgorzata Pekalska. *Dissimilarity representations in pattern recognition*. Tese de Doutoramento, Delft University of Technology, Delft, The Netherlands, 2005.

[Pol06] Robi Polikar. Pattern recognition. *Wiley Encyclopedia of Biomedical Engineering*, 2006.

[RCH+72] L. Rabiner, J. Cooley, H. Helms, L. Jackson, J. Kaiser, C. Rader, R. Schafer, K. Steiglitz, e C. Weinstein. Teminology in digital signal processing. *Audio and Electroacoustics, IEEE Transactions on*, 20(5):322–337, Dec 1972. ISSN 0018-9278.

[RN95] Stuart J. Russell e Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., 1995. ISBN 013103805.

[Ron97] Simon Ronald. Distance functions for order-based encodings. In *ICEC '97: Proceedings of 1997 IEEE International Conference on Evolutionary Computation*, páginas 49–54. IEEE, Indianapolis, IN, USA, 1997. ISBN 0-7803-3949-5.

[Sel80] Peter H. Sellers. The theory and computation of evolutionary distances: Pattern recognition. *J. Algorithms*, 1(4):359–373, 1980.

[Sil04] Flávio Roberto Silva. *Uma Abordagem para Detecção de Outliers em Dados Categóricos*. Tese de Mestrado, Universidade Estadual de Campinas, Campinas, SP, Brasil, fev 2004.

[Ska08] Matthew Adam Skala. *Aspects of Metric Spaces in Computation*. Tese de Doutoramento, University of Waterloo, Waterloo, Ont., Canada, Canada, 2008.

[Sko04] Tomáš Skopal. *Metric Indexing in Information Retrieval*. Tese de Doutoramento, University of Ostrava, June 2004.

[Sko07] Tomáš Skopal. Unified framework for fast exact and approximate search in dissimilarity spaces. *ACM Trans. Database Syst.*, 32(4):29, 2007. ISSN 0362-5915.

[Smi02] Steven W. Smith. *The Scientist & Engineer's Guide to Digital Signal Processing*. California Technical Pub., 2002. ISBN 0966017633.

[SP04] Salvador S. e Chan P. Fastdtw: Toward accurate dynamic time warping in linear time and space. Relatório técnico, Florida Institute of Technology, Melbourn, 2004. Appeared in KDD Workshop on Mining Temporal and Sequential Data.

[TTSF00] Caetano Traina Jr., Agma Traina, Bernhard Seeger, e Christos Faloutsos. Slim-Trees: High performance metric trees minimizing overlap between nodes. *Lecture Notes in Computer Science*, 1777:51–65, 2000.

[vL04] Urike von Luxburg. *Statistical Learning with Similarity and Dissimilarity Functions*. Tese de Doutoramento, Technical University of Berlin, Berlin, Germany, 2004.

[Yu05] Jian Yu. General c-means clustering model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1197, August 2005.