Graduação em Ciência da Computação

Edson de Melo Neto

APRENDIZADO DE MÁQUINA APLICADO À ACELERAÇÃO DA COMPUTAÇÃO DE PROFUNDIDADE FUNCIONAL EM DADOS UNIDIMENSIONAIS

Trabalho de Graduação

Universidade Federal de Pernambuco
secgrad@cin.ufpe.br
www.cin.ufpe.br/~secgrad

RECIFE

2025

Universidade Federal de Pernambuco
Centro de Informática
Graduação em Engenharia da Computação

Edson de Melo Neto

APRENDIZADO DE MÁQUINA APLICADO À ACELERAÇÃO DA COMPUTAÇÃO DE PROFUNDIDADE FUNCIONAL EM DADOS UNIDIMENSIONAIS

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Orientador: *Nivan Roberto Ferreira Júnior*

RECIFE
2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Melo Neto, Edson de.
Aprendizado de máquina aplicado à aceleração da computação de profundidade funcional em dados unidimensionais / Edson de Melo Neto. - Recife, 2025.
55 p. : il., tab.

Orientador(a): Nivan Roberto Ferreira Junior
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Engenharia da Computação - Bacharelado, 2025.
Inclui referências.

1. Aprendizagem de máquina. 2. Visualização de dados. I. Ferreira Junior, Nivan Roberto. (Orientação). II. Título.

000 CDD (22.ed.)

EDSON DE MELO NETO

Aprendizado de Máquina Aplicado à Aceleração da Computação de Profundidade Funcional em Dados Unidimensionais

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de bacharel em Engenharia da Computação.

Aprovado em: 31/07/2025

BANCA EXAMINADORA

_____

Prof. Nivan Roberto Ferreira Junior (Orientador)

Universidade Federal de Pernambuco

_____

Prof. Paulo G. S. da Fonseca (Examinador Interno)

Universidade Federal de Pernambuco

À memória dos meus padrinhos: Sônia e Milton.

Agradecimentos

Aos meus pais, Luiza e Roger, e aos meus irmãos, Alice e Victor, pelo apoio incondicional e por estarem comigo desde o começo, incentivando a minha curiosidade e me encorajando a traçar meu próprio caminho.

À minha namorada, Gabi, por me incentivar e ajudar sempre, por me ouvir e me entender independente de qualquer coisa.

Aos meus avós: Núbia, Jario, Ezinete e Edson, por sempre me incentivarem a seguir atrás dos meus objetivos e por todo o apoio dado durante toda a minha vida.

Aos meus amigos, Vitória, Tiago, Davi, Léo, Gabmei, Costa e Luisinho por transformarem diversos momentos de tensão e nervosismo durante a graduação em momentos de alegria e risadas.

Ao meu professor e orientador Nivan, por trabalhar comigo neste projeto e em muitos outros durante a graduação, bem como por estar sempre disposto a me ajudar, guiar e dar conselhos sempre que necessário.

Ao Centro de Informática e todos os seus funcionários e colaboradores por propiciarem um ambiente proveitoso para a construção de todo o meu aprendizado e para o desenvolvimento deste trabalho.

There are places I remember
All my life though some have changed
Some forever not for better
Some have gone and some remain
All these places had their moments
With lovers and friends I still can recall
Some are dead and some are living
In my life I've loved them all

—THE BEATLES

Resumo

Atualmente, uma quantidade nunca antes vista de dados são gerados diariamente em diferentes domínios. Uma das formas mais comuns de representar esses dados é através de séries temporais. Ao longo do tempo, diferentes formas de visualização dessas séries foram propostas, como gráficos de linhas e de densidade. Entretanto, à medida que o número de dados aumenta, extrair padrões e *outliers* deles se torna um desafio. Com o objetivo de superar esse obstáculo, uma possível ferramenta são as chamadas profundidades de dados funcionais, que permitem ordenar esses dados com respeito à centralidade de uma distribuição. Entretanto, os diversos métodos propostos para o cálculo dessas medidas de profundidade são custosos tanto em relação ao tempo necessário para o cálculo quanto em relação à memória utilizada. Dessa forma, o presente trabalho tem como objetivo propor uma nova abordagem baseada em aprendizado de máquina para a computação aproximada da profundidade de banda modificada, dando suporte à análise interativa de grandes coleções de dados de forma eficiente. Para atingir esses objetivos, realizamos experimentos comparando a abordagem proposta com a abordagem clássica do cálculo de profundidade de banda e com outro método de aproximação utilizando estruturas de resumo estatístico, no que diz respeito à performance e ao custo de memória necessário. Assim, percebemos que os resultados obtidos mostram que o método proposto neste estudo tem uma performance melhor do que os métodos clássicos e um custo de memória consideravelmente menor do que a outra aproximação, mesmo com o aumento dos dados. Ademais, o método proposto apresenta vantagens em relação à adição de novas séries no conjunto de dados, não sendo necessário o ré-calculo. Por fim, utilizamos as aproximações obtidas em uma técnica de visualização que nos permite ver a distribuição das séries e quais delas tiveram um comportamento que pode possivelmente classificá-las como *outliers*.

Palavras-chave: Visualização de dados, Aprendizagem de Máquina, Profundidade de dados funcionais, *Data Sketches*

Abstract

Nowadays, an unprecedented amount of data is generated daily across various domains. One of the most common ways to represent this data is through time series. Over time, different visualization techniques for these series have been proposed, such as line plots and density plots. However, as the amount of data increases, extracting patterns and outliers becomes a challenge. To overcome this obstacle, one possible tool is the so-called functional data depths, which allow ordering data with respect to their centrality in a distribution. Nevertheless, the various methods proposed for computing these depth measures are costly, both in terms of computation time and memory usage. Therefore, the present work aims to propose a new machine learning based approach for the approximate computation of the modified band depth, providing support for the efficient interactive analysis of large data collections. To achieve these objectives, we conducted experiments comparing the proposed approach with the classical method of computing band depth and with another approximation method, in terms of performance and memory usage. The results indicate that the method proposed in this study outperforms classical methods and requires significantly less memory than the other approximation method, even as the data scale increases. Furthermore, the proposed method presents advantages when adding new series to the dataset, as it does not require recalculation. Finally, we applied the obtained approximations in a visualization technique that enables us to observe the distribution of the curves and identify which ones may be classified as outliers.

Keywords: Data visualization, Machine Learning, Functional Data Depth, Data Sketches

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

IQR	*Inter-Quartile Range*
PDF	Profundidade de Dados Funcionais
BD	*Band Depth*
MBD	*Modified Band Depth*
TX_1	*dataset* de corridas de táxis em 2012
TX_{100}	*dataset* de corridas de táxis nos anos 2012 com cada corrida copiada 100 vezes

Lista de Algoritmos

Sumário

h

1

Introdução

Vivemos numa era em que toneladas de dados são gerados diariamente em diferentes setores da sociedade. Por exemplo, dados são coletados e gerados na medicina, mobilidade urbana, economia, entre outros (MANYIKA, 2011). Uma das principais formas de representação desses dados é através de séries temporais, em que valores são medidos e amostrados em intervalos discretos ao longo do tempo. Esse formato de registro de dados é amplamente adotado por analistas em diferentes áreas, como finanças, climatologia, entre outras (FULCHER; LITTLE; JONES, 2013); sendo representados usualmente através de gráfico de linhas.

Apesar da grande quantidade de dados disponível, extrair *insights* deles é uma tarefa complexa. Além disso, especificamente ao analisarmos gráficos de linhas, à medida que o número de séries aumenta, torna-se evidente a ineficiência de métodos tradicionais de visualização, devido a um problema conhecido como *overplotting*. A partir da Figura 1.1, podemos observar a dificuldade de se identificar padrões e extrair informações do gráfico. Nesse cenário, as profundidades de dados funcionais (PDFs) surgem como ferramentas para exploração dessas grandes coleções de dados. Dentre elas, destacam-se a profundidade de banda (BD), profundidade de banda modificada (MBD) e a profundidade extremal (LÓPEZ-PINTADO; ROMO, 2009; NARISETTY; NAIR, 2016). Contudo, o custo computacional requerido para o cálculo dessas profundidades é alto, o que inviabiliza sua aplicação direta em sistemas interativos. Para contornar o alto custo computacional para o cálculo da BD e MBD, diferentes abordagens foram propostas. SUN; GENTON; NYCHKA (2012) propuseram utilizar uma matriz de *rankings* pré-calculada que reduz a complexidade temporal do cálculo exato, entretanto adiciona um custo computacional associado à memória. Já LIMA (2019) propôs um método baseado em uma estrutura de sumarização de quantis, o *t-digest* (DUNNING; ERTL, 2019), possibilitando a rápida aproximação das profundidades. Mais recentemente, HAN et al. (2024) exploraram a utilização de redes neurais para a aceleração da computação de medidas de profundidade funcional.

Assim, faz-se necessário explorar, desenvolver e avaliar métodos eficientes para o cálculo aproximado de profundidades funcionais de dados, a fim de possibilitar sua utilização em aplicações voltadas para visualização interativa de múltiplas séries temporais. Para isso, exploramos as princpais noções de PDFs, com foco na BD e MBD. Assim, implementamos e

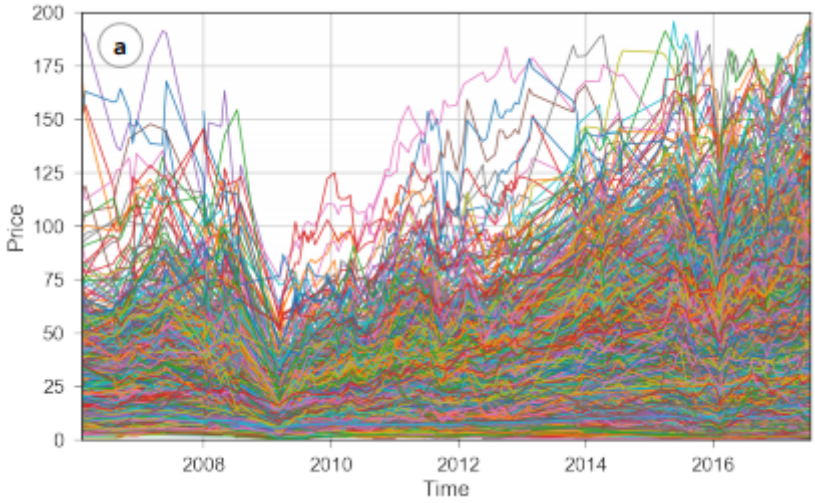


Figura 1.1: Exemplo de *overplotting* em conjunto de dados sobre valor de ações na bolsa ao longo dos anos. Fonte: MORITZ; FISHER (2018).

comparamos cinco métodos distintos para a computação aproximada da BD e da MBD. São eles: (i) o método baseado em uma matriz de *rankings* (SUN; GENTON; NYCHKA, 2012), (ii) o método que utiliza o *t*-digest (DUNNING; ERTL, 2019), (iii) o método baseado no *kll* (KARNIN; LANG; LIBERTY, 2016), (iv) o método utilizando a estrutura de sumarização *la_vector* proposta por BOFFA; FERRAGINA; VINCIGUERRA (BOFFA; FERRAGINA; VINCIGUERRA, 2021), e (v) o método proposto neste trabalho, que emprega técnicas de aprendizagem de máquina. Nosso objetivo é comparar esses métodos quanto à acurácia, tempo de execução e uso de memória, buscando identificar suas principais vantagens e limitações.

2

Trabalhos Relacionados

Neste Capítulo, apresentaremos uma revisão de artigos relacionados aos temas abordados em nosso trabalho.

2.1 Computação Aproximada de Profundidade de Dados Funcionais

LÓPEZ-PINTADO; ROMO (2009) propuseram em seu trabalho uma das mais clássicas noções de profundidade de dados, a profundidade de banda (BD). Essa noção de profundidade de dados tem como objetivo, dado um conjunto de funções observáveis, ordená-las com base em sua centralidade. Para isso, para cada série, é calculado quantas bandas a contêm, sendo uma banda definida por um par de séries do conjunto de dados, como visto na Figura 2.1, a banda formada pelas séries y_1 e y_3 em cinza. Entretanto, a profundidade de banda é muito suscetível a ruídos; qualquer pequena alteração numa série pode levar a nenhuma banda a conter, como visto na Figura 2.1, a série y_4 não é contida por nenhuma banda. Assim, um *ranking* de funções fracamente definido é gerado, o que não é de interesse quando queremos classificar um conjunto de funções observáveis. Além disso, como veremos no Capítulo 3, o cálculo da BD exige um alto custo computacional, tornando-o muito ineficiente para a aplicação em ferramentas de visualização interativa de dados.

No mesmo trabalho, LÓPEZ-PINTADO; ROMO (2009) apresentaram a profundidade de banda modificada (MBD). Essa outra noção de profundidade foi proposta visando solucionar o problema da suscetibilidade da profundidade de banda a ruídos. Ao invés de fazer uma análise binária, se uma série está contida ou não numa banda, calcula-se a proporção da série que está contida na banda. Essa pequena mudança faz grande diferença no ranqueamento definido do conjunto de dados. Entretanto, assim como a profundidade de banda, o cálculo da profundidade de banda modificada se mostra muito ineficiente. No capítulo Capítulo 3, mostraremos com detalhes o cálculo de ambas as profundidades.

Posteriormente, SUN; GENTON; NYCHKA (2012) propuseram uma abordagem aproximada para o cálculo da profundidade de banda e da profundidade de banda modificada. A fim

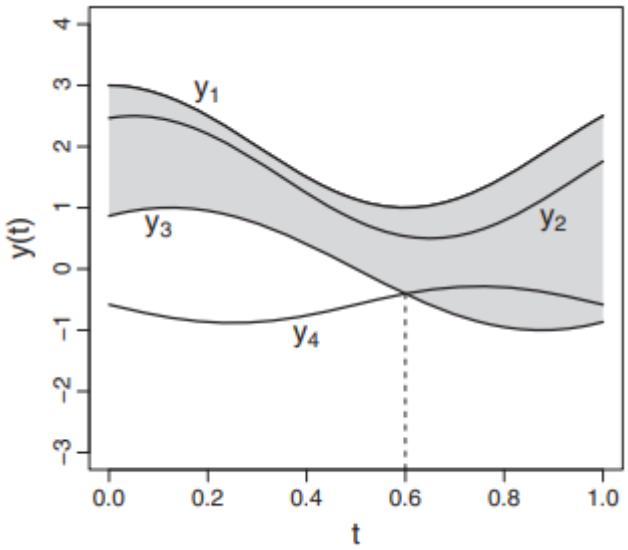


Figura 2.1: Exemplo de conjunto de dados, a região cinzenta representa a banda formada pelas séries y_1 e y_3. Fonte: SUN; GENTON; NYCHKA (2012)

.

de acelerar o cálculo de ambas as profundidades, os autores utilizaram uma matriz de *rankings* pré-calculada. Essa matriz tem dimensões p *x* n, sendo n o número de séries e p a resolução de uma série, ou seja, o número de pontos que a define. Para cada linha da matriz, temos o ranking em que cada série se encontra para um dado ponto. Assim, conseguimos para uma série, calcular quantas séries estão sempre acima, e quantas séries estão sempre abaixo dela dado os *rankings* máximos e mínimos de uma série. Logo, é possível calcular a profundidade de banda e a profundidade de banda modificada de forma aproximada e mais eficiente. No entanto, essa solução envolve a utilização de memória extra para a realização dos cálculos. Dessa forma, à medida que o número de séries aumenta, esse *overhead* de memória pode se tornar uma dificuldade.

Mais recentemente, HAN et al. (2024) propuseram uma nova abordagem para computação de profundidade para *boxplots* de superfície utilizando aprendizagem profunda. O objetivo dos autores é acelerar o cálculo da profundidade funcional usada em *boxplots* para visualização de incertezas em conjuntos de dados 2D que variam no tempo. Para isso, a partir da Figura 2.2, podemos ver que os autores utilizaram uma arquitetura do tipo *encoder* e um *decoder*, ambos construídos utilizando *multilayer perceptrons* (MLP), que são redes neurais que possuem múltiplas camadas de neurônios.

2.2 Análise Visual de Dados Temporais

Mais do que nunca, os dados surgem como um tópico central em nossas vidas. Entretanto, a análise de grandes volumes de dados espaciais e temporais se torna uma tarefa complexa

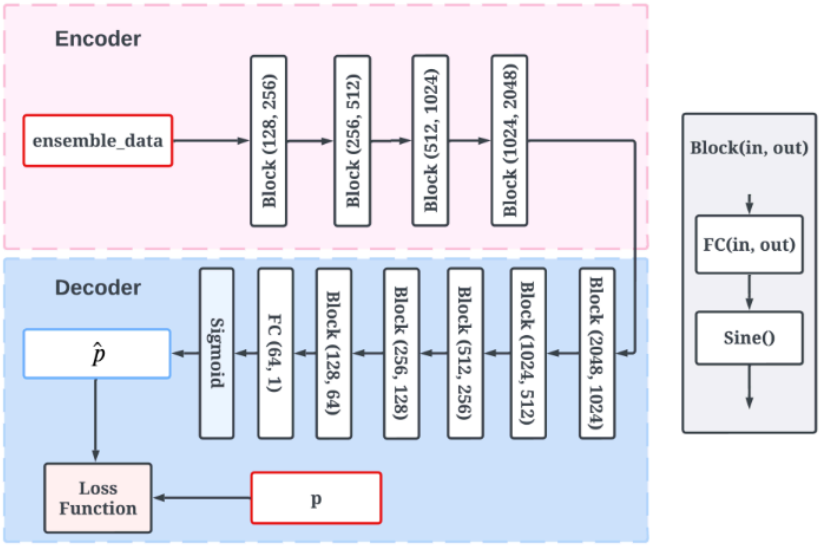


Figura 2.2: Ilustração da rede neural construída utilizando *multilayer perceptrons* (MLPs) com funções de ativações senoidais. Fonte: HAN et al. (2024).

e desafiadora devido à alta dimensionalidade e à presença de ruído. Essas dificuldades podem dificultar a extração de padrões e *insights* significativos, essenciais para uma tomada de decisão eficiente em diversas áreas, como geoinformática, monitoramento ambiental e análise de fenômenos dinâmicos. Adicionalmente, métodos clássicos de visualização de séries temporais costumam representá-las como linhas em gráfico de linhas (HOCHHEISER; SHNEIDERMAN, 2004). Esses métodos provêm ferramentas para realização de consulta de forma dinâmica, sendo possível realizar o filtro do conjunto de dados de acordo com algum padrão definido. No entanto, à medida que a quantidade de séries cresce, os gráficos de linhas sofrem de problemas como *overplotting*, como visto na Figura 1.1, e baixa performance em operações de consulta.

Para reduzir a confusão na visualização, soluções baseadas em densidade de séries temporais foram propostas.

LAMPE; HAUSER propuseram definir uma função de densidade tratando as séries como objetos geométricos contínuos ao invés de amostras discretas, expandindo a ideia de *kernel density estimation* para séries (LAMPE; HAUSER, 2011b). O objetivo dos autores foi desenvolver um método para visualizar um conjunto de séries temporais em espaços 2D, permitindo ao usuário entender distribuições espaciais e suas variações em grandes coleções de dados. Como podemos ver um exemplo na Figura 2.3, a ideia central é analisar um conjunto de séries a partir da densidade. Em cada ponto, podemos analisar a densidade de séries através de um histograma, por exemplo. Para validação, os autores testaram essa abordagem em um conjunto de dados de trajetórias de furacões. Os resultados obtidos demonstraram que utilizar uma abordagem baseada em KDEs ajudou a preservar a estrutura e a fornecer uma distribuição clara das curvas. Em suma, o método conseguiu captar a variabilidade espacial em grandes conjuntos de dados e identificar tendências dominantes de forma eficaz. Contudo, o método limita-se a utilizar conjuntos de dados 2D, sendo complexa a tarefa de analisar conjuntos de dados com

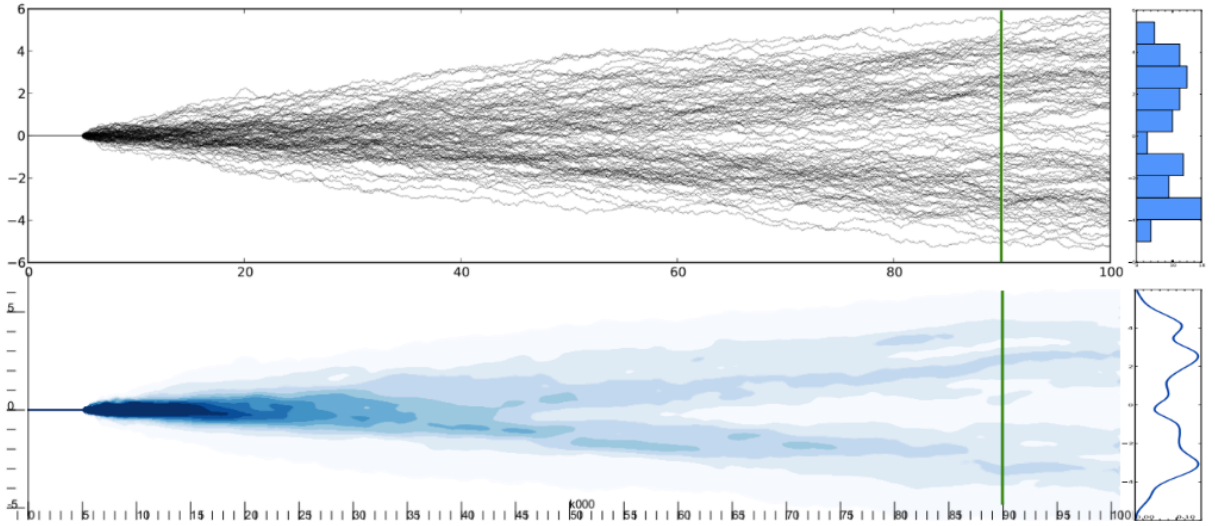


Figura 2.3: 100 séries aleatórias. O gráfico de cima mostra as séries temporais com uma transperência menor, enquanto o de baixo mostra a densidade. O histograma e o 1D KDE da direita mostram todas as amostras na linha verde no ponto x = 90. Fonte: LAMPE; HAUSER (2011a).

mais dimensões. Além disso, a interpretabilidade do valor de densidade pode ser um desafio para usuários menos experientes.

Visando mitigar o *overplotting*, ZHAO et al. (2011) propuseram o ChronoLenses, uma ferramenta que permite ao usuário, de maneira intuitiva, explorar, manipular e entender séries temporais em diferentes escalas e níveis de detalhamento. O ChronoLenes é uma técnica para exploração interativa de dados baseada em lentes. Essas lentes possibilitam ao usuário do sistema dar "*zooms*" temporais e comparar diferentes intervalos de tempos, através de agregação e transformação de dados dentro de uma região selecionada. Para validação da ferramenta, os autores conduziram um estudo qualitativo onde os participantes deveriam completar tarefas propostas. Esse estudo foi realizado com 10 participantes, incluindo cientistas de dados e experts em visualização de dados. O ChronoLenses mostrou-se uma ferramenta poderosa e flexível para a exploração interativa de séries temporais. Contudo, podemos destacar algumas limitações apresentadas pelo ChronoLenses como uma alta curva de aprendizado para usuários menos experientes, a falta de suporte para dados multivariados complexos e também a falta de escalabilidade visual à medida que o número de lentes aumenta.

Já no trabalho de LIU et al. (2022), os autores introduziram o Multivariate Time Series Visualization (MTV), um sistema que utiliza diversos tipos de visualização a fim de permitir ao usuário explorar sequências de eventos de forma eficiente. O MTV suporta tarefas como comparação, clusterização e identificação de padrões de forma escalável e intuitiva. Para isso, ele é composto de diferentes visualizações como visão de sequências, padrões, clusters e alinhamento. Essas visões têm como objetivo trazer uma visão geral, resumos estatísticos, mineração de padrões, agrupamento de padrões e permitir alinhamento e comparação temporal. Para validação foram realizados estudos de caso com usuários, incluindo experts em diferentes

domínios a fim de demonstrar a utilização do MTV para identificação de padrões, detecção de anomalias e geração de hipóteses. Assim, a ferramenta se mostrou importante para análise visual temporal de sequência de eventos. Ela traz a integração de várias visualizações, permitindo escalabilidade e uma afiada exploração dos dados. Todavia, uma grande limitação apresentada pelo MTV é o seu alto custo computacional para o tratamento dos dados, um fator que dificulta a utilização da ferramenta para grandes conjuntos de dados.

Em seu trabalho, ZHAO et al. (2021) propuseram uma nova abordagem para exploração interativa de grandes coleções de séries temporais em tempo real chamada KD-Box. Esse método busca preservar tendências importantes no conjunto de dados, permitir a procura de similaridades de forma eficiente e funcionar em diferentes níveis de granularidade. O KD-Box é uma abordagem baseada em uma árvore KD utilizando segmentos de linha, que indexa dados de séries temporais em segmentos de linha que representam tendências. Além disso, é uma estrutura fundamentada em uma busca eficiente utilizando regras de poda na árvore KD. Os experimentos realizados pelos autores compararam o tempo de construção da estrutura de indexação e a performance das consultas, demonstrando escalabilidade para datasets com até centenas de milhões de pontos. O KD-Box suportou interatividade, escalabilidade e exploração precisa de séries temporais utilizando uma nova abordagem de árvore KD baseada em segmentos de linha. Além disso, a abordagem atingiu notáveis ganhos de performance tanto na operação de indexação quanto na execução de consultas. Porém, a ferramenta desenvolvida apresenta limitações como a perda de informações em séries temporais altamente não-lineares e a introdução de erros de aproximação durante a fase de pré-processamento.

Mais recentemente, abordagens utilizando técnicas e modelos de aprendizagem profunda demonstraram potencial para exploração visual interativa de dados. Isso pode ser observado no trabalho de ZHOU et al. (2023). Utilizando *deep learning*, os autores desenvolveram um framework incorporando séries temporais em um espaço de baixa dimensão, com o objetivo de extrair *insights* desses dados. Para isso, eles propuseram o Time-series Embedding Network (TENET), um modelo treinado de forma autossupervisionada a partir de uma arquitetura encoder-decoder. Dessa forma, o encoder mapeia as séries temporais a serem analisadas em vetores compactos, enquanto o decoder tem a função de reconstruir a série original a partir da versão compactada. No trabalho, os autores realizaram análises quantitativas e qualitativas utilizando datasets variados como sensores de movimento e eletrocardiogramas. Esse estudo concluiu que o TENET conseguiu preservar eficientemente a semântica das séries temporais, conseguindo extrair padrões dos dados e também detectar *outliers*. Entretanto, um desafio comum relacionado a modelos de aprendizagem profunda é a baixa interpretabilidade e escalabilidade visual, característica também observada no TENET.

Similarmente, HAN et al. (2022) aplicaram técnicas de aprendizagem profunda para acelerar o cálculo de métodos tradicionais de visualização de dados variantes no tempo. Esses métodos são ineficientes devido a altos custos computacionais e à redundância entre timestamps. Os autores propõem uma rede neural para estimar resultados do clássico método Proba-

bilistic Marching Cubes (PÖTHKOW; WEBER; HEGE, 2011). O modelo proposto é treinado utilizando dados pré-computados através da forma naïve, além de dados como médias e desvio padrão referentes às estimativas. O modelo proposto foi avaliado tanto em dados sintéticos quanto em conjuntos de dados reais, verificando-se que o custo computacional obtido foi entre duas a três ordens de magnitude menor do que o método tradicional, enquanto manteve-se uma alta acurácia na predição. Dessa forma, o método proposto trouxe uma abordagem efetiva, diminuindo o custo computacional dos cálculos e mantendo uma alta acurácia na predição. Entretanto, algumas limitações são observadas no modelo proposto, tais como um custo de treinamento computacionalmente alto.

3

Referencial Teórico

O presente capítulo fornece mais detalhes a respeito da definição das diferentes noções de profundidade de banda abordadas por este trabalho, bem como sobre os algoritmos propostos na literatura para o cálculo dessas profundidades. Além disso, apresentamos ferramentas para visualização exploratória de dados funcionais. Por fim, apresentaremos estruturas de dados utilizadas para resumo estatístico.

3.1 Noções de Profundidade para Dados Funcionais

Inicialmente, para facilitar a compreensão das noções de profundidade, utilizaremos a seguinte notação:

- y: Uma **série temporal**.
- $Y = \{y_0, y_1, \ldots, y_{n-1}\}$: Um **conjunto de n séries temporais**.
- n: O **número total de séries** no conjunto Y.
- p: A **resolução de cada série**, ou seja, o número de pontos discretos que a definem em um intervalo de tempo.
- $y_k(l)$: O **valor da série y_k no ponto l** (para $l \in \{0, 1, \ldots, p-1\}$).
- $B(y_i, y_j)$: A **banda** definida pelas séries y_i e y_j, que é a região entre as duas curvas, ou seja, para cada ponto l, $B(y_i, y_j)(l) = [\min(y_i(l), y_j(l)), \max(y_i(l), y_j(l))]$.
- $y_k \subseteq B(y_i, y_j) \iff \forall l \in 0, \ldots, p-1, \quad \min(y_i(l), y_j(l)) \leq y_k(l) \leq \max(y_i(l), y_j(l))$
- $I\{\cdot\}$: A **função indicadora**, que retorna 1 se a condição dentro das chaves for verdadeira e 0 caso contrário.

3.1.1 Profundidade de Banda

A profundidade de banda BD foi introduzida por LÓPEZ-PINTADO; ROMO (2009). Essa medida de profundidade utiliza representações gráficas para ordenar e classificar séries temporais com base em sua centralidade, permitindo a identificação eficaz de valores atípicos e a organização de séries dos casos mais centrais aos mais extremos. Dado um conjunto de séries $Y = \{y_0, y_1, y_2, ..., y_{n-1}\}$, sendo n a quantidade de séries e p a resolução de cada séries, ou seja, o número de pontos que definem a série, temos que:

$$BD(y_k) = \binom{n}{2}^{-1} \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} I\{y_k \subseteq B(y_i, y_j)\} \tag{3.1}$$

A equação 3.1 define a profundidade de banda de uma série em relação a conjunto de séries Y, sendo $B(y_i, y_j)$ a banda definida pelas séries y_i e y_j, e I a função indicadora, ou seja, a função que indica se a série y_k pertence à banda definida anteriormente.

Como visto, o cálculo da BD envolve examinar, para cada série, quantas bandas formadas por um par de séries do conjunto de dados a envolve. Assim, até uma pequena alteração em um ponto específico de uma série pode resultar em nenhuma banda a envolvendo.

Como é o caso da série y_4 na Figura 2.1 que pertence apenas parcialmente à banda ilustrada formada pelas séries y_1 e y_3. Além disso, dependendo da instabilidade do conjunto de dados, o número de bandas que irão conter completamente uma série pode ser muito pequeno, resultando em muitas séries com o mesmo valor de profundidade.

Como consequência, a ordenação obtida após o cálculo da BD pode ter muitas séries com o mesmo valor de profundidade, o que torna o ranking fracamente definido.

Algoritmo 1 Algoritmo *check_curve* para checar se uma série está completamente contida na banda definida por duas séries.

Input: Três séries y_1, y_2, y_3 de tamanho p
Output: 1 se $y_1 \subseteq B(y_2, y_3)$, 0 caso contrário

```
for i = 0 to p − 1 do
    a ← y1.values[i]
    b ← min(y2.values[i], y3, values[i])
    c ← max(y2.values[i], y3, values[i])
    if (a < b ∨ a > c) then
        return 0
    end if
end for
return 1
```

O Algoritmo 2 mostra como é feito o cálculo da profundidade de banda dado um conjunto de Y séries de resolução p. Além disso, vemos o Algoritmo 1, que é utilizado como subrotina do Algoritmo 2 para checar, dada três séries y_1, y_2 e y_3, se a banda $B(y_2, y_3)$ definida pelas séries y_2 e y_3 contém a série y_1.

Analisando o Algoritmo 2, que é a implementação direta da definição proposta por LÓPEZ-PINTADO; ROMO (2009), podemos ver que nele temos três *for loops* aninhados. Os dois *loops* mais internos são usados para definir uma banda, enquanto que o primeiro *loop* define a série a ter a sua profundidade calculada. Além disso, vemos que o Algoritmo 1, no pior caso, itera por todos os pontos de cada série. Por isso, temos uma complexidade temporal da ordem de $O(n^3 p)$, sendo n o número de séries do conjunto Y e p a resolução de uma série.

Algoritmo 2 Algoritmo para calcular as profundidades de bandas de um conjunto de séries.

Input: Um array de séries Y com n séries de tamanho p
  for $i = 0$ to $n-1$ **do**
    $Y[i].bandDepth = 0.0$
  end for
  for $k = 0$ to $n-1$ **do**
    for $i = 0$ to $n-2$ **do**
      for $j = i+1$ to $n-1$ **do**
        $Y[k].bandDepth \leftarrow Y[k].bandDepth + check\_curve(Y[k], Y[i], Y[j])$
      end for
    end for
  end for
  $nchoose2 \leftarrow \binom{n}{2}$
  for $i = 0$ to $n-1$ **do**
    $Y[i].bandDepth \leftarrow Y[i].bandDepth/nchoose2$
  end for

3.1.2 Profundidade de Banda Modificada

No mesmo trabalho, LÓPEZ-PINTADO; ROMO (2009) introduziram a noção da profundidade de banda modificada (MBD). Como visto anteriormente, a profundidade de banda (BD) sofre por ser muito sensível a ruídos. Dessa forma, a profundidade de banda modificada se propõe a superar essa limitação.

Dado um conjunto de séries $Y = \{y_0, y_1, y_2, ..., y_{n-1}\}$, sendo n a quantidade de séries e p a resolução de cada série, ou seja, o número de pontos que definem a série, temos que:

$$MBD(y_k) = \frac{1}{p}\binom{n}{2}^{-1} \quad \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} \sum_{l=0}^{l=p-1} I\{y_k(l) \subseteq B(y_i(l), y_j(l))\} \tag{3.2}$$

A Equação 3.2 define a profundidade de banda modificada de uma série em relação a conjunto de séries Y, sendo $y_k(l)$ a série k do conjunto Y amostrada no ponto l, $B(y_i(l), y_j(l))$ a banda definida pelas séries y_i e y_j no ponto l, e I a função indicadora, ou seja, a função que indica se a série $y_k(l)$ pertence à banda definida anteriormente.

Como visto, o cálculo da MBD envolve examinar, para cada série e cada possível banda, a proporção da série que está envolvida pela banda. Com essa modificação, temos uma definição

de uma banda mais robusta, pois uma pequena alteração na série terá uma menor influência no cálculo da profundidade.

Algoritmo 3 Algoritmo *check_proportion* para checar a proporção de uma série que está contida na banda definida por duas séries.

Input: Três séries y_1, y_2, y_3 de tamanho p
Output: A proporção da série y_1 que está contida na banda $B(y_2, y_3)$

```
cnt ← 0
for i = 0 to p − 1 do
    a ← y1.values[i]
    b ← min(y2.values[i], y3, values[i])
    c ← max(y2.values[i], y3, values[i])
    if (a < b ∨ a > c) then
        cnt ← cnt + 1
    end if
end for
return cnt/p
```

Algoritmo 4 Algoritmo para calcular as profundidades de banda modificada de um conjunto de séries.

Input: Um array de séries Y com n séries de tamanho p

```
for i = 0 to n − 1 do
    Y[i].modifiedBandDepth = 0.0
end for
for k = 0 to n − 1 do
    for i = 0 to n − 2 do
        for j = i + 1 to n − 1 do
            proportion ← check_proportion(Y[k], Y[i], Y[j])
            Y[k].modifiedBandDepth ← Y[k].modifiedBandDepth + proportion
        end for
    end for
end for
nchoose2 ← (n 2)
for i = 0 to n − 1 do
    Y[i].modifiedBandDepth ← Y[i].modifiedBandDepth/nchoose2
end for
```

O Algoritmo 4 mostra como é feito o cálculo da profundidade de banda modificada dado um conjunto de Y séries de resolução p. Além disso, temos o Algoritmo 3, que é utilizado como subrotina do Algoritmo 4 para checar, dada três séries y_1, y_2 e y_3, qual proporção da banda $B(y_2, y_3)$ definida pelas séries y_2 e y_3 contém a série y_1.

Analisando o Algoritmo 4, que é a implementação direta da definição proposta por LÓPEZ-PINTADO; ROMO (2009), podemos ver que nele temos três *for loops* aninhados. Os dois *loops* mais internos são usados para definir uma banda, enquanto que o primeiro *loop* define a série a ter a sua profundidade calculada. Além disso, vemos que o Algoritmo 3 itera

por todos os pontos de cada série. Por isso, temos uma complexidade temporal da ordem de $O(n^3 p)$, sendo n o número de séries do conjunto Y e p a resolução de uma série.

3.1.3 Cálculo da Profundidade de Banda Utilizando Matriz de *Rankings*

Dessa forma, SUN; GENTON; NYCHKA (2012) propuseram uma abordagem que utiliza uma matriz de *rankings* visando acelerar o cálculo da profundidade de banda e da profundidade de banda modificada. Essa matriz é pré-calculada e utilizada para realizar o cálculo aproximado das profundidades.

Seja M a matriz de dados $p \times n$ representando n séries contendo p pontos. Para construir a matriz de rankings, primeiro ordenamos os n *data points* de cada linha da matriz do menor para o maior e em seguida salvamos os *rankings* $r_{ij} \in \{1,...,n\}$ na matriz de *ranking* $R_{p\times n}$. Como podemos ver no exemplo a seguir:

$$M = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} \begin{array}{c} \begin{array}{ccc} y_0 & y_1 & y_2 \end{array} \\ \begin{pmatrix} 5 & 10 & 4 \\ 12 & 3 & 8 \\ 7 & 9 & 11 \\ 13 & 6 & 2 \end{pmatrix} \end{array} \rightarrow M_{ordered} = \begin{pmatrix} (4,y_2) & (5,y_0) & (10,y_1) \\ (3,y_1) & (8,y_2) & (12,y_0) \\ (7,y_0) & (9,y_1) & (11,y_2) \\ (2,y_2) & (6,y_1) & (13,y_0) \end{pmatrix} \rightarrow R = \begin{pmatrix} 2 & 3 & 1 \\ 3 & 1 & 2 \\ 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$$

Com a matriz R construída, fixando a série y_j, podemos calcular quantas séries estão sempre abaixo e quantas séries estão sempre acima da série y_j. Assim a Equação (3.3) representa quantas séries estão sempre abaixo da série y_j e a Equação (3.4) representa quantas séries estão sempre acima acima da série y_j.

$$n_b(j) = \min_{1\leq i\leq p}(r_{ij}) - 1 \tag{3.3}$$

$$n_a(j) = n - \max_{1\leq i\leq p}(r_{ij}) \tag{3.4}$$

Com esses valores é possível obter o valor de profundidade de banda a partir de:

$$BD(y_j) = \binom{n}{2}^{-1} n_a(j) n_b(j) + (n-1) \tag{3.5}$$

na Equação acima, $n-1$ representa o número de bandas que tem y_j na borda. Da mesma forma que a profundidade de banda original, a profundidade de banda modificada pode ser calculada usando a abordagem proposta por SUN; GENTON; NYCHKA (2012), modificando as Equações (3.3) e (3.4), da seguinte forma:

$$n_b(i,j) = r_{ij} - 1 \tag{3.6}$$

$$n_a(i,j) = n - r_{ij} \tag{3.7}$$

obtemos, em determinado ponto i, quantas séries estão acima e quantas séries estão abaixo de y_j.

$$MBD(y_j) = \binom{n}{2}^{-1} \left(\frac{\sum_{1 \leq i \leq p} n_a(i,j) n_b(i,j)}{p} + (n-1) \right) \tag{3.8}$$

Assim, a Equação (3.8) acima nos dá a profundidade de banda modificada para a série y_j. O termo $n_a(i)n_b(i)$ representa, no ponto i, a quantidade de bandas que contêm a série, dessa forma o termo $\frac{\sum_{1 \leq i \leq p} n_a(i,j) n_b(i,j)}{p}$ nos dá a proporção de tempo que a série está contida nas bandas que não contêm ela na fronteira, enquanto $(n-1)$ denota o número de bandas em que a série está contida na fronteira.

Continuando o exemplo anterior, vamos realizar o cálculo da profundidade de banda modificada para o conjunto $Y = \{y_0, y_1, y_2\}$ representado na matriz M. Para isso, precisamos utilizar as Equações (3.6) e (3.7). Assim, temos que:

	$y_{j=0}$	$y_{j=1}$	$y_{j=2}$
$n_a(1,j)$	1	0	2
$n_a(2,j)$	0	2	1
$n_a(3,j)$	2	1	0
$n_a(4,j)$	0	1	2
$n_b(1,j)$	1	2	0
$n_b(2,j)$	2	0	1
$n_b(3,j)$	0	1	2
$n_b(4,j)$	2	1	0
MBD	0.75	0.83	0.75

Tabela 3.1: Cálculo da Profundidade de Banda Modificada para o conjunto Y representado pela matriz M.

Podemos analisar o custo computacional da solução proposta por SUN; GENTON; NYCHKA (2012) tanto para a profundidade de banda quanto para a profundidade de banda modificada.

Inicialmente, vemos que temos um cálculo em comum para ambos os métodos, que envolve o cálculo da matriz R. Para isso, vemos que precisamos ordenar cada uma das p linhas da matriz M. Então, se escolhermos o algoritmo *quicksort*, que tem complexidade $O(nlog(n))$, para ordenarmos os dados, ficamos com uma complexidade temporal de $O(pnlog(n))$. Para a profundidade de banda, a partir das Equações (3.3) e (3.4), precisamos calcular o máximo de cada coluna, respectivamente. Esse cálculo envolve iterar sobre todos os elementos de cada coluna, tendo um custo temporal total de $O(np)$.

Assim, para o cálculo da profundidade de banda temos uma complexidade temporal da ordem $O(nlog(n) + np)$. Enquanto que para a profundidade de banda modificada, temos uma complexidade temporal da ordem $O(nlog(n))$. Em relação à implementação *naïve*, temos uma melhoria expressiva na complexidade temporal. Entretanto, temos um aumento em relação à memória utilizada, tendo um custo de memória da ordem de $O(np)$, memória utilizada para

armazenamento da matriz de *rankings*.

3.2 Ferramentas para Visualização Exploratória de Dados Funcionais

3.2.1 *Functional Boxplot*

Atualmente, em diversos setores da sociedade, como a medicina e a meteorologia, observamos um crescimento sem precedentes de dados funcionais, onde cada observação é definida por uma série ou função. Entretanto, cada método de visualização existente ou utiliza redução de dimensionalidade ou não consegue representar fielmente a natureza dos dados.

Dessa maneira, SUN; GENTON (2011) propuseram uma nova abordagem para o desenvolvimento de uma ferramenta de visualização direta, robusta e intuitiva voltada para dados funcionais. Os autores apresentaram o functional boxplot, uma extensão do boxplot clássico para dados funcionais (HYNDMAN; SHANG, 2010). Esse método desenvolvido deve ser capaz de sumarizar séries, identificar tendências centrais e variabilidade, e detectar outliers nos dados.

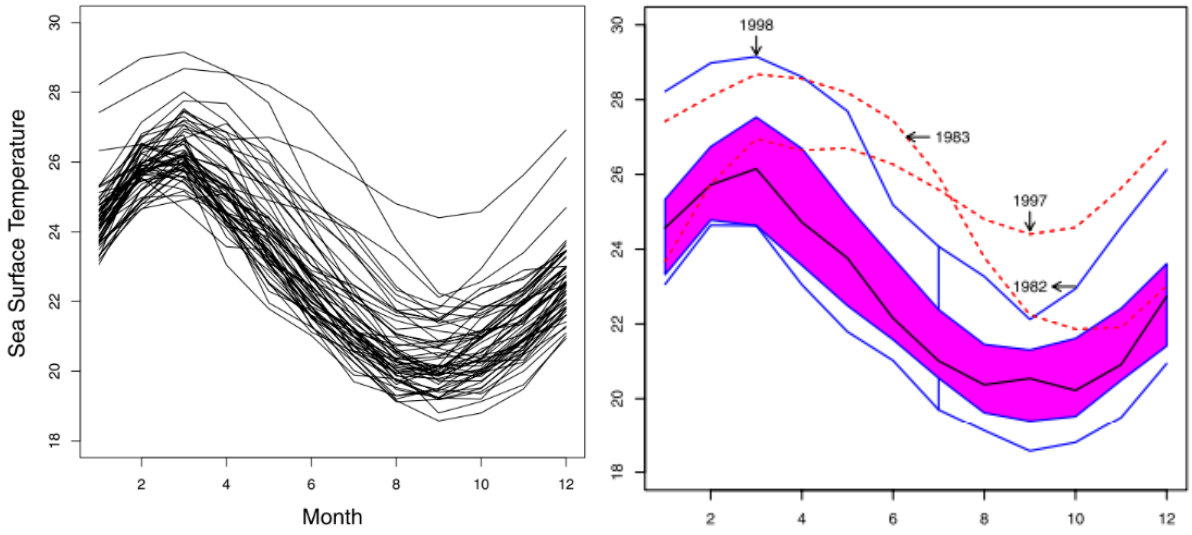
O conceito de centralidade em dados funcionais refere-se à posição relativa de uma série dentro de um conjunto, com base em sua similaridade com as demais. De maneira análoga à mediana em dados univariados, a série mais central representa o comportamento típico do conjunto. Para quantificar essa centralidade, SUN; GENTON (2011) utilizam medidas de profundidade, como a profundidade de banda (BD) e a profundidade de banda modificada (MBD), que permitem ordenar as séries da mais central à mais extrema. Analogamente ao *boxplot* clássico, o *functional boxplot* apresenta:

- O envelope da região 50% central, ou seja, 50% dos dados com maior profundidade, o que corresponde ao *Inter-Quartile Range* (IQR);
- A série mediana, ou seja, a série com maior BD/MBD;
- O envelope máximo sem *outliers* que é calculado inflando a amplitude do envelope IQR por um fator de 1,5. Dessa forma, qualquer série fora desse envelope é considerada um possível *outlier*.

Na Figura 3.1b, temos o envelope em rosa que define a região 50% central, em preto a série mediana do conjunto de dados e em azul a região do envelope máximo sem *outliers*. Destacado tracejado em vermelho, temos as séries consideradas *outliers* do conjunto.

3.2.2 *Surface Boxplot*

Analogamente à seção anterior, vemos um crescimento de dados complexos como imagens que representam, por exemplo, ressonâncias magnéticas cerebrais e modelos climáticos.



(a) (b)

Figura 3.1: (a) Dados mensais de temperatura da superfície do mar, medidos em graus Celsius, sobre o Oceano Pacífico tropical centro-leste, no período de 1951 a 2007. (b) O boxplot funcional da temperatura da superfície do mar, com curvas azuis representando os envelopes e uma série preta indicando a série mediana. As séries tracejadas em vermelho são os candidatos a outliers detectados pela regra de 1,5 vezes a região central de 50%. Fonte: SUN; GENTON (2011).

Entretanto, métodos tradicionais se mostram insuficientes para visualização e detecção de outliers em conjuntos de dados de alta dimensionalidade. Por isso, faz-se necessário um método intuitivo, robusto e eficiente para sumarizar e explorar tais conjuntos de dados.

Assim, em seu trabalho, GENTON et al. (2014) introduziram os boxplots de superfície, uma extensão de boxplots funcionais que busca sumarizar e visualizar dados de imagem 3D, identificando superfícies centrais, detectando outliers e destacando a variabilidade dos dados. O método proposto, chamado profundidade de volume modificada (MVD), é usado para tratar imagens como superfícies 3D. Para esse método, a superfície média é a de maior profundidade. Envelopes internos e externos (análogos aos quartis em boxplots 1D) são construídos e outliers são detectados baseados em simples regras. É possível observar um exemplo de boxplot de superfície na Figura 3.2. Além disso, foram realizados experimentos utilizando dois conjuntos de dados distintos. Primeiro, o método foi utilizado para identificar atividades cerebrais anormais em um conjunto de ressonâncias magnéticas cerebrais (OASIS). Segundo, o método foi aplicado para identificar anomalias climáticas em dados climáticos. De modo geral, os boxplots de superfícies foram eficientes em representar conjuntos de dados de imagens, identificando imagens outliers e representativas do conjunto.

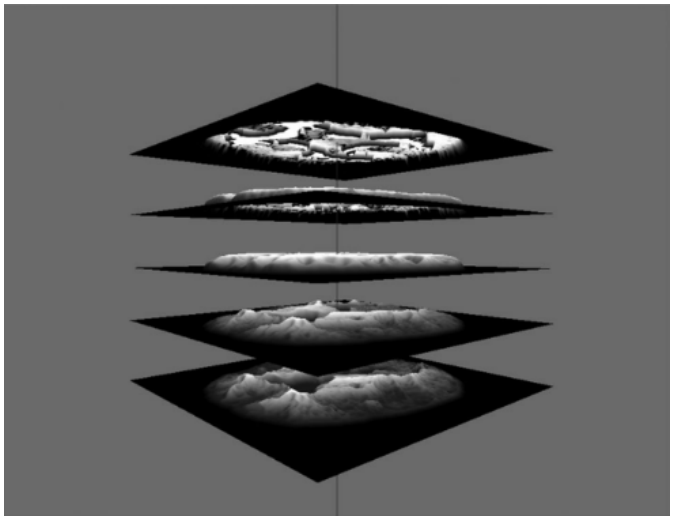


Figura 3.2: Exemplo de *surface boxplot*, a imagem é codificada como um campo de altura. A superfície mediana é o campo de altura central, ladeado pelos envelopes interno e externo. Fonte: GENTON et al. (2014)

.

3.3 Estruturas de Dados para Resumo Estatístico

3.3.1 T-Digest

Almejando diminuir o custo de memória usado pela abordagem proposta por SUN; GENTON; NYCHKA (2012), utilizaremos a estrutura de resumo estatítisco *t*-digest para realizar o cálculo aproximado da BD e MBD. O *t*-digest foi proposto no trabalho de DUNNING; ERTL (2019) com o objetivo de computar aproximações estatísticas de ordem com alta precisão e de maneira *on-line*, ou seja, sem a necessidade de reter todas as amostras ou aguardar até que todos os dados tenham sido processados, problemas encontrados nos algoritmos clássicos vistos anteriormente.

A estrutura do *t*-digest é baseada nos chamados centróides. Cada centróide representa um agrupamento de dados reais, sendo preservado a média e a quantidade de pontos sendo sumarizados. Quando essa estrutura recebe um conjunto de dados, a definição dos centróides é feita através do agrupamento dos elementos em subsequências de tamanhos variados, seguindo uma estratégia flexível. Essa estratégia funciona dando prioridade para quantis nas extremidades da distribuição dos dados, tornando menor os centróides próximos ao início ou ao fim, assim é possível ter uma estimativa mais precisa das regiões extremas, tornando mais precisa a detecção de *outliers*.

O processo de construção de um *t*-digest é feito a partir de um vetor de valores, que são os elementos do conjunto passado para a estrutura. Periodicamente, após a inserção de um determinado número de dados, os centróides são comprimidos, mantendo o tamanho da estrutura de dados gerenciável. Para estimar um quantil, o *t*-digest utiliza a posição relativa dos

centróides e a distribuição acumulada dos valores através de interpolação.

3.3.2 KLL

Similarmente à estrutura *t*-digest, visando diminuir o custo de memória adicional usada pela abordagem proposta por SUN; GENTON; NYCHKA (2012), vamos usar a estrutura de dados *kll* para o cálculo aproximado da profundidade de banda e profundidade de banda modificada. O *kll* foi proposto no trabalho de KARNIN; LANG; LIBERTY (2016) para computar aproximações estatísticas de ordem com alta precisão e de maneira *on-line*, ou seja, sem a necessidade de reter todas as amostras ou aguardar até que todos os dados tenham sido processados, problemas encontrados nos algoritmos clássicos vistos anteriormente.

A estrutura *kll* utiliza uma hierarquia de "containers" de capacidades variáveis para o armazenamento dos dados. Supondo que tenhamos H containers no total que são indexados pela sua altura $h \in \{1,...,H\}$. O peso dos itens de altura h é dado por $w_h = 2^{h-1}$ e a capacidade do container por k_h. Dessa forma, quando um container h atinge sua capacidade máxima, primeiramente seus itens são ordenados, e após isso, os elementos nas posições pares ou ímpares são escolhidos. Os elementos não escolhidos são descartados, e os outros são colocados no container $h+1$ que tem peso $w_{h+1} = 2 * w_h$.

Por fim, podemos consultar o *rank* de um elemento x, sendo ele o número de elementos estritamente menores do que x. Isso pode ser feito passando uma vez por cada container e é definido da seguinte forma:

$$rank(x) = \sum_{l} 2^l * rank_l(x) \tag{3.9}$$

Similarmente ao mostrado para o *t*-digest, podemos calcular as medidas de profundidade de banda através dos Algoritmos 5 e 6, considerando que o vetor E de *data sketches* é um vetor de *kll*s. Essa abordagem utilizando o *kll* traz diversas vantagens. Primeiramente, o fato de poder ser calculada de forma *on-line*. Além disso, o *overhead* de memória, mostrado no trabalho de SUN; GENTON; NYCHKA (2012), diminui significativamente ao utilizarmos a estrutura *kll*, fato esse que é demonstrado na seção de experimentos.

3.3.3 *LA Vector*

Recentemente, BOFFA; FERRAGINA; VINCIGUERRA (2021) propuseram uma abordagem inovadora para a representação eficiente de dicionários ordenados, concentrando-se nas operações de classificação e seleção. A ideia central dos autores é a utilização de segmentos para aproximar um conjunto de pontos no plano cartesiano com ênfase na compressão eficiente do dicionário. Dessa forma, seja $S = \{x_1, x_2, ..., x_n\}$ um conjunto de pontos, transformamos esse conjunto em pontos (i, x_i) no plano cartesiano, e tentamos aproximar essa curva em segmentos de reta. Cada segmento $s_j = (r_j, \alpha_j, \beta_j)$ cobre um intervalo de índices e aproxima os valores

Algoritmo 5 Algoritmo para achar *rankings* máximos e mínimos das séries utilizando E

Input: Um conjunto de séries Y com n séries de tamanho p e E um vetor de *data sketches*

```
for j = 0 to p − 1 do
    for i = 0 to n − 1 do
        rank ← (E[j].findQuantile(Y[i].values[j]) ∗ n) + 1
        if rank > Y[i].maxRank then
            Y[i].maxRank = rank
        end if
        if rank < Y[i].minRank then
            Y[i].minRank = rank
        end if
    end for
end for
```

usando a fórmula:

$$f_j(i) = (i - r_j) * \alpha_j + \beta_j \tag{3.10}$$

Entretanto, essa aproximação é imperfeita, então precisamos de um ajuste, tal que:

$$x_i = \lfloor f_j(i) \rfloor + C[i] \tag{3.11}$$

Assim, a estrutura total, chamada "la_vector" (*Linear Approximation Vector*), armazena os segmentos s_j, os valores de correção $C[i]$ e algumas estruturas auxiliares para navegação eficiente.

Por exemplo, se quisermos comprimir um conjunto $S = \{3,6,10,15,18,22\}$ com erro máximo de valor 3, podemos realizar essa compressão utilizando apenas um segmento de reta $s_1 = (1,5,0)$ com erros $C = [3,1,0,0,-2,-3]$. Se realizarmos uma consulta utilizando a Equação 3.11 para $i = 5$, temos:

$$x_5 = (5-1) * 5 + 0 + (-2) = 18$$

Analogamente à utilização da matriz de rankings e aos *data sketches*, podemos utilizar o la_vector como uma estrutura auxiliar para o cálculo da profundidade de banda e profundidade de banda modificada. Assim, como mostrado anteriormente, podemos calcular as medidas de profundidade de banda através dos Algoritmos 5 e 6, sendo E um vetor de *la_vector*s.

Algoritmo 6 Algoritmo para achar proporções das séries de um conjunto usando E

Input: Um conjunto de séries Y com n séries de tamanho p, E um vetor de *data sketches*
Output: $proportion$, um array de tamanho n

$match \leftarrow matrix_{pn}$
$proportion \leftarrow n * [0.0]$
for $j = 0$ to $p-1$ **do**
  for $i = 0$ to $n-1$ **do**
    $rank \leftarrow (E[j].findQuantile(Y[i].values[j]) * n) + 1$
    $n_a \leftarrow n - rank$
    $n_b \leftarrow rank - 1$
    $match[j][i] \leftarrow n_a * n_b$
  end for
end for
for $i = 0$ to $n-1$ **do**
  $sum \leftarrow 0$
  for $j = 0$ to $p-1$ **do**
    $sum \leftarrow sum + match[j][i]$
  end for
  $proportion[i] \leftarrow sum/p$
end for
return $proportion$

4

Método de Aproximação Proposto

Neste capítulo, apresentamos o método proposto para a aproximação da profundidade de banda (BD) e profundidade de banda modificada (MBD) utilizando técnicas de aprendizagem de máquina, com foco no modelo Extra Trees.

A motivação para essa abordagem reside no alto custo computacional envolvido no cálculo tradicional da BD e MBD, especialmente em contextos com grandes volumes de séries temporais ou com alta resolução temporal. O objetivo é empregar um modelo preditivo que, treinado com um subconjunto reduzido de dados, consiga inferir com boa precisão os valores de profundidade associados a cada série. Dessa forma, reduz-se drasticamente o tempo de processamento sem comprometer a qualidade da estimativa.

O modelo utilizado é o Extremely Randomized Trees (Extra Trees), que pertence à família dos algoritmos baseados em árvores de decisão. Diferentemente do Random Forest, o Extra Trees não realiza amostragem dos dados (bootstrapping) e escolhe os pontos de divisão de maneira totalmente aleatória. Essas características promovem maior diversidade entre as árvores do conjunto, o que reduz o risco de sobreajuste (*overfitting*) e o custo computacional. A implementação é realizada por meio da biblioteca **PyCaret** [1], que oferece uma interface de alto nível para experimentação e avaliação de modelos de aprendizado de máquina.

4.1 Fluxo da Aplicação

O processo proposto para a aproximação das medidas de profundidade é composto por cinco etapas principais:

- Pré-processamento dos dados, incluindo normalização e preparação das variáveis de entrada e saída;
- Seleção de um subconjunto de séries que serão utilizadas para o treinamento e teste do modelo;
- Divisão das séries selecionadas em dois conjuntos: Treinamento e Teste;

[1] `https://pycaret.org/`

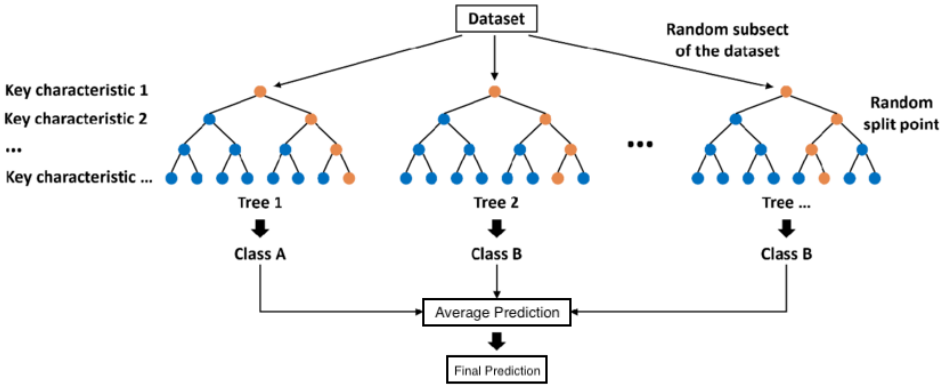


Figura 4.1: Esquema do modelo Extra Trees utilizado para regressão.

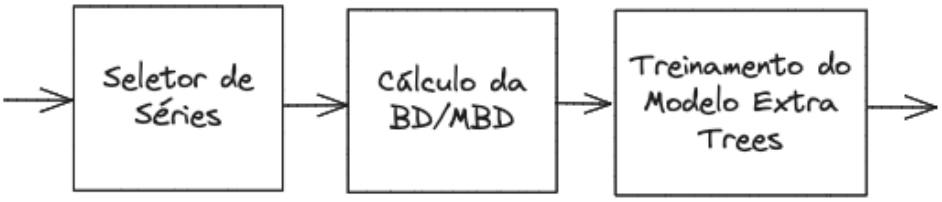


Figura 4.2: Fluxo do método proposto utilizando o modelo Extra Trees para previsão das profundidades de banda.

- Cálculo da profundidade de forma *naïve* para o conjunto selecionado;
- Treinamento e aplicação do modelo Extra Trees para regressão dos valores de profundidade.

A Figura 4.1 ilustra, de forma esquemática, a estrutura do modelo Extra Trees utilizado. Além disso, a Figura 4.2 representa o fluxo completo proposto descrito acima, desde a etapa de seleção de séries até o treinamento do modelo proposto.

4.2 Pré-processamento e Treinamento

Cada série é representada por um vetor de valores inteiros. Esses vetores são utilizados como entrada para o modelo Extra Trees. Antes do treinamento, os dados são normalizados utilizando o `StandardScaler`[2], que aplica a normalização Z-score (média zero e desvio padrão um). Essa padronização garante que todos os atributos contribuam de forma equilibrada para o processo de aprendizado e facilita a comparação entre experimentos. Os valores de saída, correspondentes às profundidades reais (BD ou MBD), também são escalonados separadamente, permitindo uma aprendizagem mais estável por parte do modelo. A divisão dos dados é feita por meio da estratégia de *holdout*, com 70% das amostras destinadas ao treinamento e 30% ao teste. Uma semente aleatória é utilizada para garantir reprodutibilidade. O modelo Extra Trees

[2]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

é configurado e treinado automaticamente utilizando o módulo `regression` da biblioteca PyCaret. Esse processo inclui a criação do conjunto de treino/teste, aplicação das transformações, treino do modelo e avaliação de desempenho com validação cruzada embutida.

4.3 Inferência e Avaliação

Após o treinamento, o modelo é utilizado para realizar inferência sobre as séries do conjunto de teste. Os valores previstos são revertidos do escalonamento e comparados com os valores reais de profundidade. A principal métrica de avaliação utilizada é o erro absoluto médio (MAE), que fornece uma medida direta da precisão das estimativas sem amplificar erros extremos.

4.4 Considerações Finais

A utilização do modelo Extra Trees se mostrou uma alternativa promissora ao cálculo exato da profundidade funcional, principalmente em contextos com restrição de tempo ou recursos computacionais. Os experimentos apresentados no Capítulo 5 demonstram a viabilidade dessa abordagem, com bons níveis de precisão e significativa redução de tempo de processamento.

5

Análise experimental

Este capítulo apresenta os experimentos realizados. Fizemos uma análise comparativa entre as estruturas de resumo estatístico apresentadas no Capítulo 3 e o método proposto nesse trabalho. Para isso, comparamos a implementação *naïve* da profundidade de banda (BD) e da profundidade de banda modificada (MBD) com os métodos aproximados mostrados e nosso método proposto. Especificamente, comparamos o método que utiliza a matriz de *rankings*, o método proposto por LIMA (2019) e o método proposto neste trabalho, utilizando uma abordagem de aprendizagem de máquina. Vale ressaltar que o método proposto por LIMA (2019) utiliza a estrutura estatística *t-digest* para realização das aproximações de profundidade. Generalizando a ideia, realizamos uma análise comparativa utilizando outros *data sketches*, o *KLL* e o *la_vector*.

5.1 Metodologia

Nesta seção iremos descrever a metodologia utilizada para os experimentos realizados visando validar o método de aproximação proposto no Capítulo 4.

5.1.1 Os Dados

Para a realização dos experimentos, utilizamos o conjunto de dados referente às corridas de táxi na cidade de Nova Iorque no ano de 2012 [1]. O conjunto de dados de táxis de Nova Iorque contém diversos atributos das corridas de táxi, dentre eles o número de passageiros, a distância, os horários de começo e término da corrida, dentre outros. A fim de construir séries temporais, para a análise experimental realizada, utilizamos a coluna que apresenta o horário de início/fim da corrida para obter coleções de séries temporais diárias (uma por dia) da quantidade de viagens em cada hora do dia, a cada 30 minutos ou a cada 15 minutos, a depender da resolução da série utilizada. Para comparação do tempo de execução, usaremos dois datasets; o primeiro que chamaremos de TX_1 representa as corridas de táxi de Nova Iorque do ano de

[1] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

2012, o segundo que chamaremos de TX_{100} é o conjunto de dados TX_1 onde cada corrida foi copiada 100 vezes, a fim de aumentarmos o tamanho do dataset.

5.1.2 Implementação

Para a realização dos experimentos, usamos três estruturas de dados principais, sendo elas o *t*-digest, *KLL* e *la_vector*. Para a estrutura *la_vector*, utilizamos a implementação dos autores BOFFA; FERRAGINA; VINCIGUERRA, feita na linguagem de programação C++, a fim de assegurar a fiel reprodução dos resultados obtidos pelos autores originais. Para as estruturas *t*-digest e *KLL*, usamos a implementação da biblioteca *Apache DataSketches*[2] implementadas na linguagem de programação C++. Optamos por essa abordagem para comparar métricas como tempo de execução e utilização de espaço entre todas essas estruturas; assim, utilizar a implementação do *la_vector* em C++ tornou essa decisão necessária. A comparação entre o método proposto e outras estruturas, como *t*-digest, *la_vector* e *KLL*, será conduzida através da avaliação de três métricas principais: tempo de execução, uso de espaço e qualidade das operações de classificação e seleção. Para cada métrica, serão estabelecidos procedimentos específicos de medição, proporcionando uma análise abrangente do desempenho relativo das estruturas em questão. Os experimentos foram realizados utilizando o conjunto de dados proveniente das corridas de táxi da cidade de Nova Iorque. A escolha desse conjunto de dados deve-se à sua disponibilidade e acessibilidade, e ao fato de que o conjunto de dados é livre de inconsistências e dados ausentes.

5.1.3 Algoritmos

Para os métodos de cálculo aproximado da profundidade de banda e profundidade de banda modificada, usamos os seguintes algoritmos propostos por LIMA (2019):

A partir do Algoritmo 5, é possível calcular a profundidade de banda (BD) assim como o método proposto por SUN; GENTON; NYCHKA, a partir da Equação 5.1.

$$\mathrm{BD}(y) = \binom{n}{2}^{-1} (y.\mathrm{minRank} - 1) * (n - y.\mathrm{maxRank}) + n - 1 \tag{5.1}$$

Além disso, podemos utilizar o Algoritmo 6 para calcular a profundidade de banda modificada como dada pela Equação 5.2.

$$\mathrm{MBD}(y) = \binom{n}{2}^{-1} \left(\frac{\sum_{i=1}^{p} (n - y[i].\mathrm{rank})(y[i].\mathrm{rank} - 1)}{p} + n - 1 \right) \tag{5.2}$$

Por fim, podemos notar em ambos os Algoritmos 5 e 6 que existe um vetor E de p estruturas de dados que são utilizadas para aproximar o *ranking* de uma série em um dado ponto;

[2] http://datasketches.apache.org/

essas estruturas podem ser qualquer uma das mencionadas anteriormente, como o *t*-digest, *kll* e *la_vector*.

5.1.4 Comparação do Custo Computacional das Implementações dos Métodos de Cálculo de Profundidade de Banda

Nesses experimentos, um dos principais objetivos foi analisar os dois principais aspectos computacionais: o tempo de execução de cada algoritmo e também o custo de memória utilizada por cada algoritmo.

5.1.5 Medindo o Desempenho de Tempo dos Algoritmos

Na medição do tempo de execução de cada algoritmo, a estratégia utilizada foi medir o tempo levado pela função que calcula cada profundidade de banda.

Para o cálculo da profundidade de banda (BD) proposta por LÓPEZ-PINTADO; ROMO, considerou-se o tempo decorrido para checar se uma banda contém uma série, dado pelo Algoritmo 1. Além disso, foi contabilizado o tempo da divisão pelo número total de bandas do conjunto, $\binom{n}{2}$, onde n é o número de séries existentes no *dataset*.

Para o cálculo da profundidade de banda modificada (MBD), também proposta por LÓPEZ-PINTADO; ROMO, considerou-se o tempo levado para checar toda série e toda banda, a porcentagem da série que está contida numa banda, dada pelo Algoritmo 4. Além disso, também foi contabilizado o tempo necessário para a divisão pelo número total de bandas do conjunto, $\binom{n}{2}$, onde n é o número de séries existentes no *dataset*.

A fim de calcular as aproximações para a BD e a MBD utilizando a matriz de rankings proposta por SUN; GENTON; NYCHKA, foi levado em consideração o tempo transcorrido para criar e montar a matriz de *rankings*, além do cálculo dos *rankings* máximos e mínimos de cada série. Já para os métodos que utilizam as estruturas de dados t-digest, kll e la_vector, foi contabilizado o tempo decorrido para o cálculo dos mínimos e máximos, e da proporção da série que está contida em cada banda. Para todos os métodos acima, utilizamos a biblioteca *C Time*[3] para mensurar o tempo decorrido. Por fim, para o modelo de aprendizagem de máquina, considerou-se o tempo necessário para o cálculo da profundidade para o subconjunto de dados selecionado, para o treinamento do modelo e para a etapa de validação e teste. Para isso, usamos a biblioteca *Time*[4] em Python.

5.1.6 Medindo o Custo de Memória dos Algoritmos

A medição do custo de memória de cada algoritmo apresentado foi feita levando em consideração a necessidade de cada um. Para o cálculo dos métodos originais, a profundidade

[3]https://cplusplus.com/reference/ctime/

[4]https://docs.python.org/3/library/time.html

de banda (BD) e a profundidade de banda modificada (MBD) não foram utilizadas nenhuma memória. Já para o método proposto por SUN; GENTON; NYCHKA, existe a necessidade de utilização de uma matriz de *rankings*, que é uma estrutura que varia de acordo com o tamanho do conjunto de dados utilizado. Nesse caso, utiliza-se uma matriz de tamanho $n \times p$, sendo n o número de séries do dataset e p a resolução da série, ou seja, o número de pontos que estão sendo analisados por série, resultando num custo de memória de $O(np)$. Já para as abordagens utilizando as estruturas de dados *t*-digest, *kll* e *la_vector*, utilizam um *t*-digest, um *kll* e um *la_vector*, respectivamente, para cada instante de tempo analisado, sendo da nossa vontade analisar como o custo de memória dessas estruturas varia de acordo com datasets de diferentes tamanhos. Para os casos acima, foi utilizada a biblioteca padrão da linguagem C++[5]. Por fim, para o modelo de aprendizagem de máquina, levou-se em consideração o tamanho do modelo serializado gerado pela biblioteca Pycaret.

5.2 Resultados Experimentais

Na seção subsequente, realizaremos uma análise comparativa entre as estruturas apresentadas e o modelo proposto de aprendizagem de máquina em relação à qualidade de aproximação, tempo de execução e uso de memória.

5.2.1 Comparação dos Resultados dos Algoritmos

Podemos observar, nas Figuras 5.1, 5.2, e 5.3, os erros relativos entre a implementação naïve da profundidade de banda (BD), os métodos de aproximação utilizados e o método proposto nesse trabalho. Vemos que os erros encontrados para os métodos utilizando t-digests, klls e la_vectors foram semelhantes nas três resoluções de séries temporais testadas. Em todos os casos, o método utilizando a matriz de rankings se saiu um pouco melhor do que os mencionados. Além disso, podemos notar que os erros obtidos foram relativamente altos; isso se deve ao fato de que a profundidade de banda é muito suscetível a ruídos, o que torna mais difícil a generalização e aproximação do conjunto de dados. Entretanto, vemos que o modelo de aprendizagem de máquina obteve um resultado quase duas ordens de grandeza melhor do que os métodos aproximados, e nota-se que a acurácia do modelo aumentou conforme a resolução da curva cresceu.

Entretanto, para a profundidade de banda modificada (MBD), o comportamento foi diferente. Para todas as resoluções, as melhores aproximações obtidas foram utilizando as estruturas *t*-digest e *la_vector*. Logo atrás, vieram as aproximações realizadas pela estrutura *kll*, e por fim, o método utilizando a matriz de rankings proposto por SUN; GENTON; NYCHKA (2012). Vemos ainda que o modelo de aprendizagem de máquina se saiu pior do que os métodos utilizando as estruturas de *t*-digest e *la_vector*, mas obteve um desempenho próximo ao método

[5] https://cplusplus.com/reference/cstdlib/

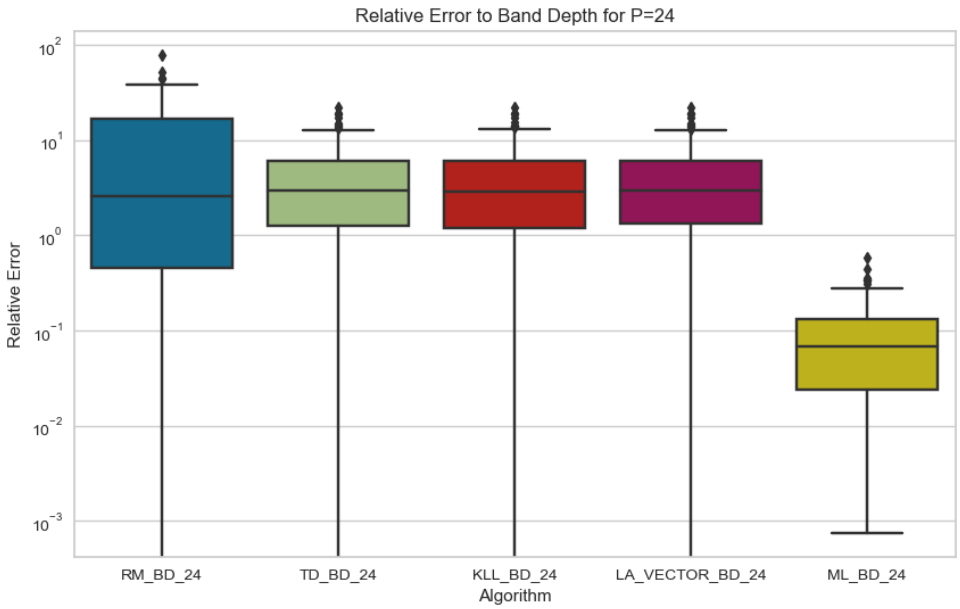


Figura 5.1: Boxplots representando a distribuição dos erros relativos dos valores de profundidade de banda obtidos por cada um dos algoritmos analisados em relação à implementação naïve da profundidade de banda para uma resolução de série igual a 24 pontos (intervalos de 1 hora) e dataset TX_1.

utilizando o *kll* e melhor do que o método que utiliza a matriz de *rankings*. Porém, vemos que em relação aos erros obtivos para a BD, todos os modelos se saíram melhor, o que nos diz que as estruturas de resumo estatístico sofreram com os ruídos nos dados, entretanto se sobresaem à medida que esses ruídos influenciam menos na medida de profundidade.

5.2.2 Comparação do Desempenho dos Algoritmos

Resolução	BD	RM_BD	TD_BD	KLL_BD	LA_VECTOR_BD	ML_BD
24	19.4 s	0.001 s	0.026 s	0.012 s	0.008 s	0.076 s
48	20.0 s	0.002 s	0.052 s	0.024 s	0.016 s	0.083 s
96	21.8 s	0.004 s	0.112 s	0.050 s	0.035 s	0.090 s

Tabela 5.1: Tempo de execução dos métodos estudados para o cálculo da profundidade de banda no dataset TX_1 variando a resolução das séries. Para o método de aprendizagem de máquina, o tempo corresponde ao treinamento e teste do modelo.

A Tabela 5.1 mostra o tempo (em segundos) - que cada um dos algoritmos levou para calcular a profundidade de banda original (BD). Podemos observar que o método naïve foi o que obteve o pior desempenho de modo geral, tendo uma complexidade temporal de $O(n^3 p)$, sendo n o número de séries no dataset e p o número de pontos a serem analisados por série. Já para os métodos de aproximação, podemos observar que o método que utiliza a matriz de rankings obteve o melhor desempenho. Em seguida, tivemos os métodos que utilizam *t*-digests, *kll*s e *la_vectors* obtendo desempenhos semelhantes, enquanto o método de aprendizagem de

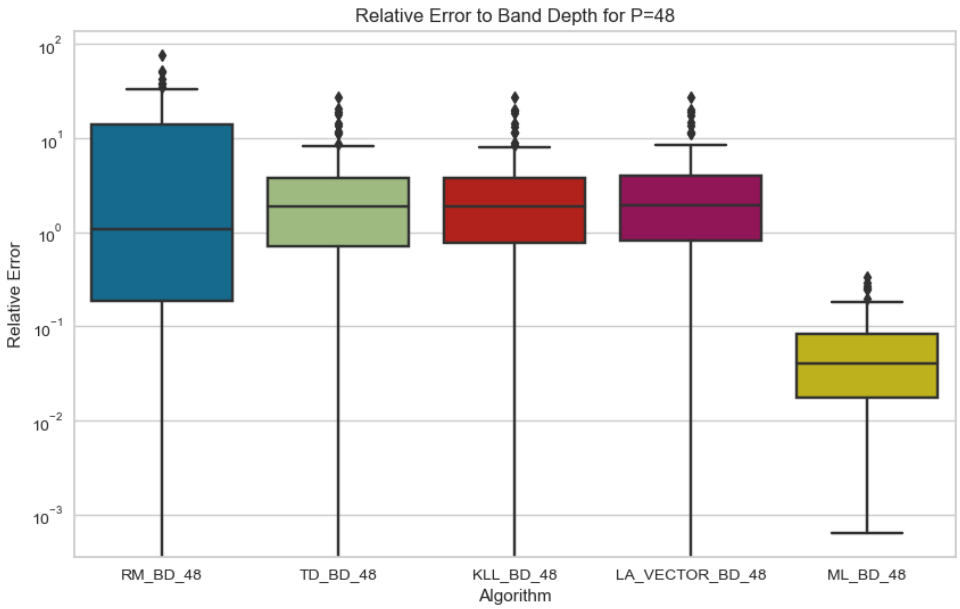


Figura 5.2: Boxplots representando a distribuição dos erros relativos dos valores de profundidade de banda obtidos por cada um dos algoritmos analisados em relação à implementação naïve da profundidade de banda para uma resolução de série igual a 48 pontos (intervalos de 30 minutos) e dataset TX_1.

Resolução	MBD	RM_MBD	TD_MBD	KLL_MBD	LA_VECTOR_MBD	ML_MBD
24	35.2 s	0.001 s	0.026 s	0.013 s	0.013 s	0.077 s
48	52.6 s	0.002 s	0.053 s	0.025 s	0.025 s	0.099 s
96	83.6 s	0.004 s	0.109 s	0.052 s	0.038 s	0.131 s

Tabela 5.2: Tempo de execução dos métodos estudados para o cálculo da profundidade de banda modificada no dataset TX_1 variando a resolução das séries. Para o método de aprendizagem de máquina, o tempo corresponde ao treinamento e teste do modelo.

máquina obteve um desempenho melhor do que o método naïve, porém pior que os outros métodos de aproximação. Além disso, a partir da Tabela 5.2, podemos observar um cenário semelhante no desempenho dos algoritmos utilizados para o cálculo da profundidade de banda modificada que notamos no cálculo da profundidade de banda.

5.2.3 Comparação do Custo de Memória dos Algoritmos

Além da medição do tempo, também medimos o custo de memória adicional para cada um dos algoritmos testados. Com isso, nosso objetivo foi de analisar o *trade-off* entre o desempenho do algoritmo e quanta memória adicional ele precisa para realizar o cálculo. As Tabelas 5.3, 5.4, 5.5 e 5.6 mostram, em megabytes, o custo de memória de cada um dos algoritmos analisados nessa etapa. Para uma melhor análise, decidimos variar a resolução das séries assim como a quantidade de séries no *dataset* a fim de investigar o comportamento dos métodos estudados. Primeiramente, podemos ver a partir da Tabela 5.3, o custo de memória do cálculo da profundidade de banda para o *dataset* TX_1. Podemos ver que o método que obteve o menor

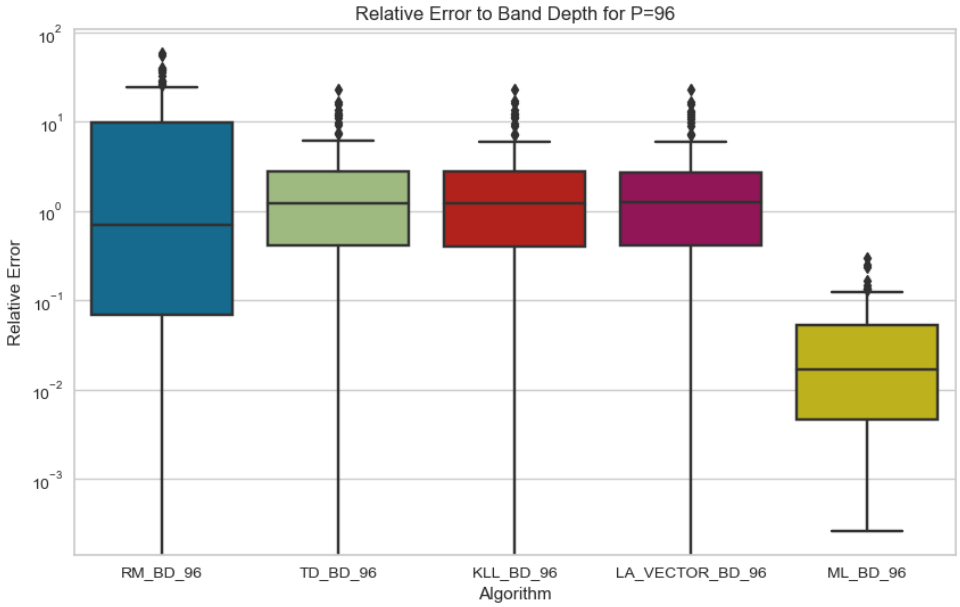


Figura 5.3: Boxplots representando a distribuição dos erros relativos dos valores de profundidade de banda obtidos por cada um dos algoritmos analisados em relação à implementação naïve da profundidade de banda para uma resolução de série igual a 96 pontos (intervalos de 15 minutos) e dataset TX_1.

custo de memória foi o *la_vector*, juntamente com o método naïve. Já o modelo de aprendizagem de máquina foi o que utilizou mais memória extra, chegando a utilizar 20 vezes mais memória que o método com melhor desempenho. Todavia, a partir da Tabela 5.4, podemos ver os benefícios do modelo de aprendizagem de máquina, pois ao aumentarmos o tamando do nosso *dataset* em 100 vezes, a quantidade de memória necessária permaneceu a mesma, enquanto que para todos os outros métodos, o custo aumentou de 15 a 70 vezes, já que esse valor é proporcional à quantidade de séries no conjunto de dados. Analogamente ao observado para o cálculo da profundidade de banda, podemos analisar o custo de memória adicional para o cálculo da profundidade de banda modificada. O resultado obtido foi semelhante ao visto anteriormente. Demonstrando os benefícios do modelo de aprendizagem de máquina em relação aos métodos de aproximação estudados à medida que o conjunto de dados cresce.

Resolução	RM_BD	TD_BD	KLL_BD	LA_VECTOR_BD	ML_BD
24	0.3 MB	0.6 MB	0.4 MB	0.2 MB	4.3 MB
48	0.4 MB	0.9 MB	0.7 MB	0.3 MB	3.6 MB
96	0.6 MB	1.7 MB	1.1 MB	0.5 MB	2.7 MB

Tabela 5.3: Custo de memória dos métodos estudados para o cálculo da profundidade de banda no dataset TX_1 variando a resolução das séries.

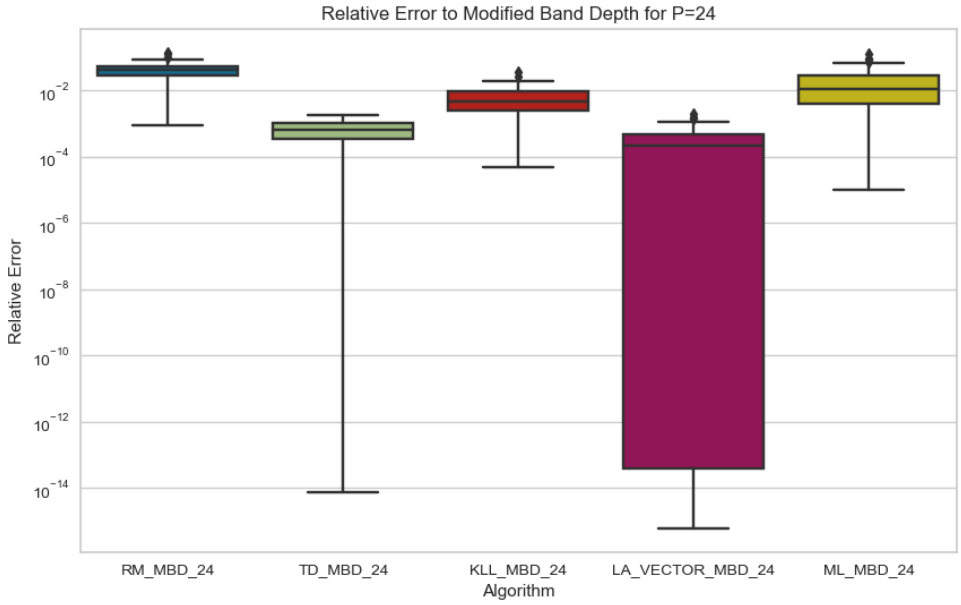


Figura 5.4: Boxplots representando a distribuição dos erros relativos dos valores de profundidade de banda obtidos por cada um dos algoritmos analisados em relação à implementação naïve da profundidade de banda modificada para uma resolução de série igual a 24 pontos (intervalos de 1 hora) e dataset TX_1.

Resolução	RM_BD	TD_BD	KLL_BD	LA_VECTOR_BD	ML_BD
24	7.8 MB	9.4 MB	8.9 MB	12.3 MB	4.3 MB
48	11.5 MB	13.5 MB	12.8 MB	20.6 MB	3.6 MB
96	18.9 MB	21.8 MB	20.8 MB	36.1 MB	2.7 MB

Tabela 5.4: Custo de memória dos métodos estudados para o cálculo da profundidade de banda no dataset TX_{100} variando a resolução das séries.

Resolução	RM_MBD	TD_MBD	KLL_MBD	LA_VECTOR_MBD	ML_MBD
24	0.2 MB	0.6 MB	0.4 MB	0.3 MB	1.3 MB
48	0.4 MB	1.0 MB	0.7 MB	0.4 MB	1.3 MB
96	0.6 MB	1.8 MB	1.2 MB	0.6 MB	1.4 MB

Tabela 5.5: Custo de memória dos métodos estudados para o cálculo da profundidade de banda modificada no dataset TX_1 variando a resolução das séries.

Resolução	RM_MBD	TD_MBD	KLL_MBD	LA_VECTOR_MBD	ML_MBD
24	7.8 MB	10.0 MB	8.8 MB	12.4 MB	4.3 MB
48	11.5 MB	14.1 MB	12.7 MB	20.2 MB	3.6 MB
96	18.9 MB	22.4 MB	20.7 MB	35.6 MB	2.7 MB

Tabela 5.6: Custo de memória dos métodos estudados para o cálculo da profundidade de banda modificada no dataset TX_{100} variando a resolução das séries.

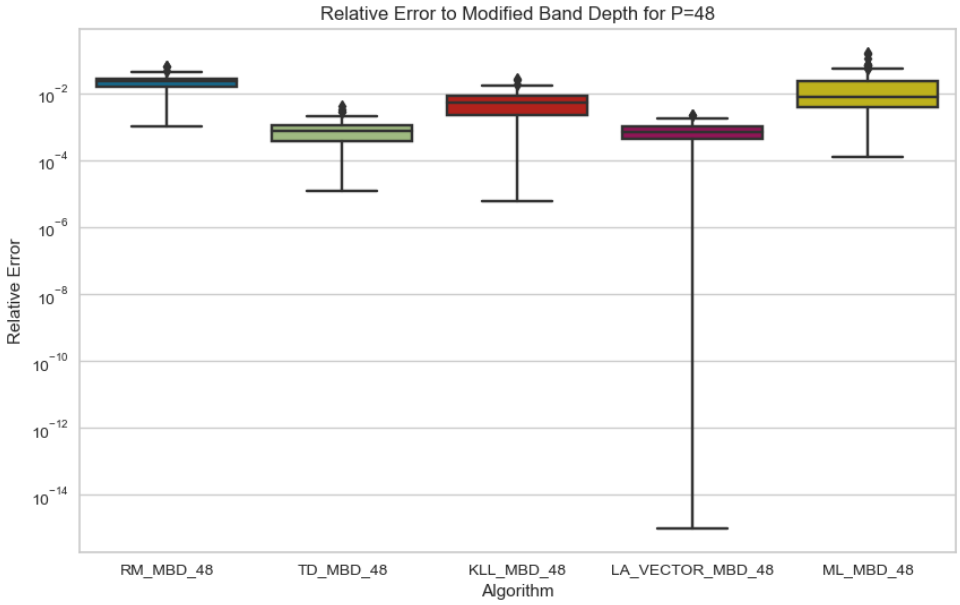


Figura 5.5: Boxplots representando a distribuição dos erros relativos dos valores de profundidade de banda obtidos por cada um dos algoritmos analisados em relação à implementação naïve da profundidade de banda modificada para uma resolução de série igual a 48 pontos (intervalos de 30 minutos) e dataset TX_1.

5.2.4 Erro Relativo do Modelo Proposto

Como discutido anteriormente, é do nosso interesse ser capaz de treinar um modelo de aprendizagem de máquina a partir de um subconjunto de séries temporais do *dataset*, mantendo um baixo erro relativo na previsão da profundidade de banda (BD) e da profundidade de banda modificada (MBD).

A Figura 5.7 apresenta os valores de erro relativo médio obtidos para diferentes combinações de parâmetros p e n, onde p representa a resolução de cada série e n representa a porcentagem do *dataset* utilizado no treinamento do modelo. Observa-se que, de forma geral, o erro relativo diminui à medida que o número de séries de treinamento aumenta, o que é esperado dado que mais informação é fornecida ao modelo.

Além disso, é possível notar que, mesmo com quantidades relativamente baixas de séries, como 30 ou 35, os erros já se mantêm em patamares bastante aceitáveis (inferiores a 0.06), indicando que o modelo proposto é eficiente e robusto mesmo com conjuntos de treinamento reduzidos.

Esses resultados mostram que o modelo de Extra Trees é capaz de aprender de maneira eficaz a profundidade funcional a partir de um subconjunto limitado de dados, o que pode ser particularmente útil em cenários onde o cálculo exato da profundidade seria computacionalmente custoso para grandes volumes de dados.

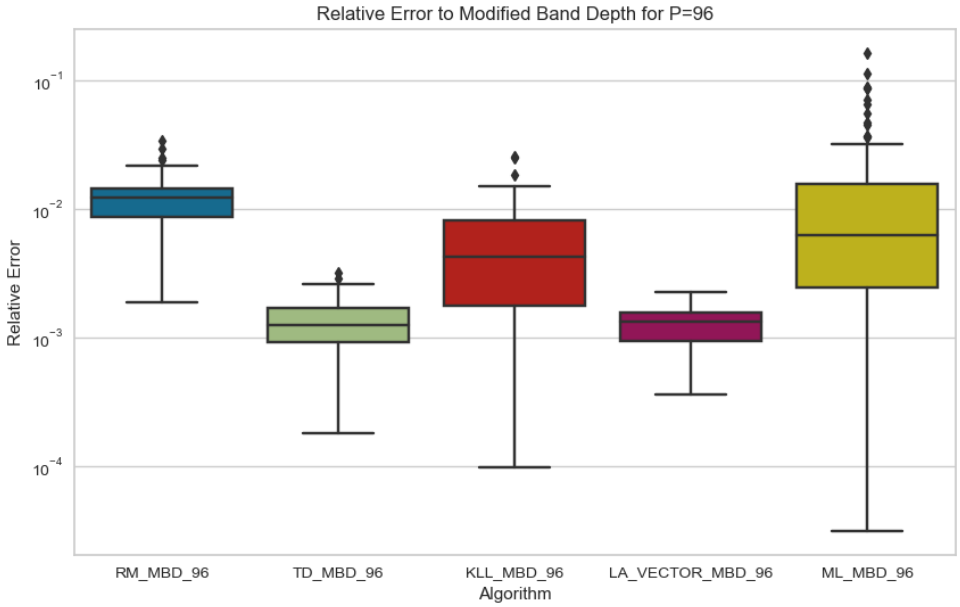


Figura 5.6: Boxplots representando a distribuição dos erros relativos dos valores de profundidade de banda obtidos por cada um dos algoritmos analisados em relação à implementação naïve da profundidade de banda modificada para uma resolução de série igual a 96 pontos (intervalos de 15 minutos) e dataset TX_1.

P	10	15	20	25	30	35	40	45	50	55	60	65	70	75
24	0.073	0.068	0.065	0.052	0.058	0.058	0.045	0.043	0.043	0.039	0.047	0.044	0.041	0.042
48	0.077	0.071	0.071	0.059	0.059	0.053	0.045	0.046	0.043	0.039	0.045	0.044	0.037	0.038
96	0.085	0.076	0.070	0.061	0.060	0.056	0.049	0.044	0.049	0.046	0.045	0.041	0.040	0.040

Figura 5.7: Comparação do erro relativo obtido pelo modelo Extra Trees em relação ao método naïve para o cálculo da profundidade de banda. Relação entre variação da resolução das séries e tamanho do dataset de treinamento (em porcentagem).

5.3 Discussão

Comparando tanto a complexidade temporal quanto o custo de memória, podemos fazer uma análise entres os quatro métodos analisados, sendo eles o método naïve, o método que utiliza a matriz de *rankings*, o método aproximado que utiliza *data sketches* e o método proposto que utiliza aprendizagem de máquina. Primeiramente, podemos observar que o método naïve, apesar de não precisar de nenhum tipo de memória, é impraticável de ser usado em aplicações que exijam uma baixa latência, devido ao seu alto custo temporal que se mostra mais evidente à medida que o conjunto de dados aumenta. Em seguida, notamos um comportamento inverso ao método naïve no método proposto por SUN; GENTON; NYCHKA (2012), que utiliza a matriz de *rankings*. Enquanto esse método tem uma complexidade temporal menor do que o método naïve, o custo de memória é da ordem de $O(np)$, proporcional à quantidade de dados analisados. Logo, esse *overhead* de memória pode ser um fator determinante para a não utilização da matriz de *rankings* em larga escala. Em contrapartida, os métodos que utilizam *data sketches* para realizar as aproximações da BD e MBD se mostraram competitivos em relação ao tempo

gasto para computação das profundidades e à memória requerida. Todavia, como observado, à medida que a quantidade de dados analisados aumenta, o custo de memória pode se tornar um desafio quando comparado ao método proposto neste trabalho. Por fim, o método proposto, que utiliza aprendizagem de máquina para realização do cálculo aproximado da BD e MBD, se mostrou uma alternativa eficiente tanto em termos de tempo quanto de memória. Em relação ao custo temporal, apresenta-se a vantagem de, uma vez treinado, o modelo não precisar ser retreinado a cada modificação no conjunto de dados, enquanto que para os outros métodos existe a necessidade da recomputação. Além disso, em relação à memória, vemos que à medida que o conjunto de dados analisados aumenta, o modelo consegue não sofrer grandes alterações na quantidade de memória utilizada, enquanto que para a matriz de *rankings* e para os *data sketches* existe um grande aumento de memória necessária.

6

Caso de Uso

Neste capítulo, mostraremos como o conceito de profundidade de banda modificada (MBD) pode ser uma poderosa ferramenta na exploração interativa de dados. Para isso, utilizaremos o conceito de *functional boxplots*, introduzido no trabalho de SUN; GENTON (2011) e explicado no Capítulo 3, calculando os valores aproximados de profundidade de banda modificada utilizando o método proposto neste trabalho no Capítulo 4. Além disso, decidimos explorar o conjunto de dados TX_1 utilizando a MBD por ser mais robusta e menos sensível a ruídos. Assim, utilizaremos as características apresentadas pelos *functional boxplots*, como IQR e detecção de *outliers*, para identificar comportamentos anômalos no nosso *dataset*. Com esse objetivo, utilizamos a implementação do *functional boxplot* da biblioteca Statsmodel [1] em Python.

Na Figura 6.1, observamos o *functional boxplot* referente às corridas de táxi da cidade de Nova Iorque do ano de 2012. Nele a banda em cinza escuro representa o IQR, ou seja, os 50% centrais dos dados em cada ponto do gráfico; além disso, a região delimitada em cinza mais claro representa o envelope máximo sem *outliers*; e a série em preto representa a série mediana, sendo a que tem o maior valor de MBD. As demais séries presentes no gráfico representam os potenciais *outliers* observados no *dataset*, sendo elas as séries temporais cujas datas podem ser observadas na legenda.

Analisando a distribuição dos dados no gráfico da Figura 6.1, podemos ver que dentre os *outliers*, a série verde, correspondente ao dia 29 de Outubro de 2012, destaca-se pelo baixo número de corridas a partir das 9hrs da manhã. Além disso, observamos o declínio na quantidade de viagens no dia 30 de Outubro de 2012, principalmente no período entre 0h e 8h. Isso pode ser atribuído ao Furacão Sandy, um ciclone tropical que atingiu a cidade de Nova Iorque durante esse período. O furacão causou inundações, apagões e ventos fortes na cidade. Além disso, aeroportos foram fechados e voos cancelados. Portanto, é razoável inferir que a quantidade de corridas registradas nesses dias foram atípicas devido ao risco associado ao deslocamento pela cidade. Do mesmo modo, também vemos o dia 25 de Dezembro, a série roxa, com um número mais baixo de viagens de táxi do que o normal, provavelmente devido às

[1] https://www.statsmodels.org/

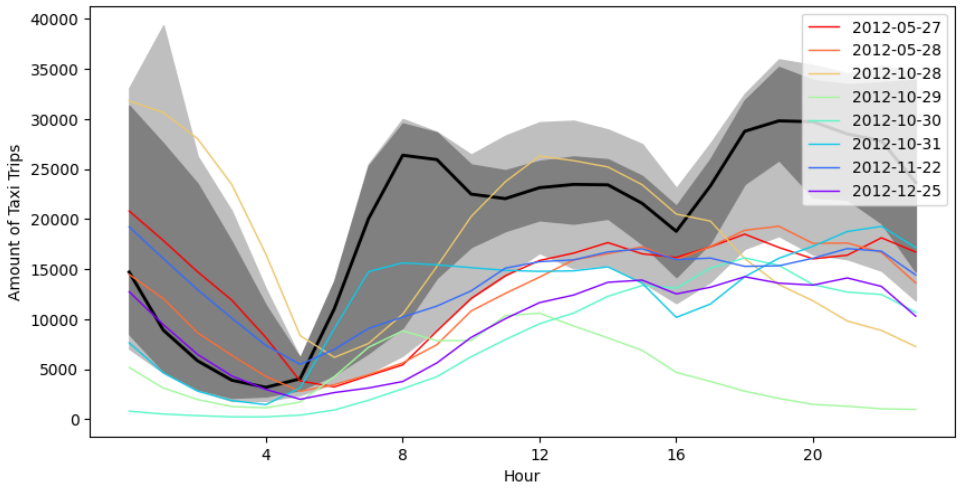


Figura 6.1: *Functional boxplot* do resultado do cálculo da profundidade de banda modificada obtido pelo algoritmo proposto no conjunto TX_1. Fonte: do autor.

festas natalinas. O dia 22 de Novembro, dia de ação de graças dos Estados Unidos, também se encontra entre as séries com menor profundidade representado pela série azul.

7

Conclusão

Vimos que a visualização de grandes coleções de séries temporais é um desafio atual. Técnicas clássicas como o gráfico de linhas sofre com o problema de *overplotting* (PLAYFAIR, 1801), mostrando impraticável a sua utilização em grandes conjuntos de dados. Dessa forma, outras abordagens foram sugeridas, como o conceito de densidade para visualização dessas séries (MORITZ; FISHER, 2018). No entanto, essa técnica se mostra limitada ao falhar em identificar anomalias em regiões densas e padrões em regiões pouco densas.

Além disso, mostramos que o conceito de PDF é uma ferramenta com muito potencial na análise de dados funcionais, permitindo a ordenação dos dados com base em medidas de centralidade. Dessas noções, destacam-se a profundidade de banda e a profundidade de banda modificada, ambas propostas por LÓPEZ-PINTADO; ROMO (2009). Entretanto, como demonstrado, ambos os métodos não são indicados para a exploração interativa de dados, devido ao seu alto custo relacionado ao tempo de computação. Assim, a fim de superar essa limitação, SUN; GENTON; NYCHKA (2012) propuseram a computação através de uma matriz de *rankings*. Todavia, essa abordagem apresenta um *overhead* de memória utilizada, tornando inviável sua utilização em grandes coleções de dados. Por fim, o método proposto por LIMA (2019) se mostrou eficiente em relação ao tempo de computação das medidas de profundidade, mas como observado, essa abordagem não escala bem para *datasets* maiores, devido a um maior custo de mémoria.

Dessa forma, apresentamos uma nova abordagem para o cálculo aproximado da BD e MBD, a qual consiste na utilização de um modelo de *machine learning* para realização das aproximações. Esse método é capaz de manter um tempo de computação competitivo em relação aos outros métodos enquanto não apresenta um elevado aumento de memória utilizada à medida que o conjunto de dados cresce; características essencias para aplicações interativas para exploração de dados.

Através dos experimentos, foi possível observar que o método proposto neste trabalho fornece uma aproximação precisa da MBD. Além disso, nosso método obteve um desempenho muito superior ao algoritmo naïve, e apesar de apresentar um desempenho inferior de tempo em relação aos outros métodos de aproximação para conjuntos de dados pequenos, torna-se

evidente sua vantagem em relação à escalabilidade. Ademais, vimos as vantagens em relação ao custo de memória utilizada em *datasets* maiores. Por fim, observamos que conseguimos obter uma boa aproximação, utilizando poucos dados para o processamento, o que é desejável para aplicações em tempo real.

Somado a isso, mostramos como a profundidade de banda modificada pode ser útil na visualização de grandes coleções de dados através de um caso de uso envolvendo o conjunto de dados referente a corridas de táxi na cidade de Nova Iorque. Através disso mostramos como esse conceito pode ser útil na identificação e visualização de séries temporais com comportamento distinto, mesmo com um grande número de séries na coleção.

Em trabalhos futuros, podemos expandir nossa abordagem para outras noções de PDF, como a profundidade extremal (NARISETTY; NAIR, 2016). Essa noção de profundidade apresenta produtos em sua computação que podem ser explorados mais a fundo, a fim de melhor compreender os dados. Isso pode ser aliado ao desenvolvimento de um *dashboard* composto por diferentes formas de visualização conectadas que permita ao usuário melhor explorar o conjunto de dados desejado. Dessa forma, a partir da computação aproximada das medidas de profundidade desejadas utilizando redes neurais, podemos construir uma ferramenta rápida e robusta para a exploração interativa de grandes coleções de dados.

Referências

BOFFA, A.; FERRAGINA, P.; VINCIGUERRA, G. A Learned Approach to Quicken and Compress Rank/Select Dictionaries. In: PROCEEDINGS OF THE WORKSHOP ON ALGORITHM ENGINEERING AND EXPERIMENTS (ALENEX), 2021. **Anais...** [S.l.: s.n.], 2021. p.46–59.

DUNNING, T.; ERTL, O. Computing extremely accurate quantiles using t-digests. **arXiv preprint arXiv:1902.04023**, [S.l.], 2019.

FULCHER, B. D.; LITTLE, M. A.; JONES, N. S. Highly comparative time-series analysis: the empirical structure of time series and their methods. **Journal of the Royal Society Interface**, [S.l.], v.10, n.83, p.20130048, 2013.

GENTON, M. G. et al. Surface boxplots. **Stat**, [S.l.], v.3, n.1, p.1–11, 2014.

HAN, M. et al. Accelerated probabilistic marching cubes by deep learning for time-varying scalar ensembles. In: IEEE VISUALIZATION AND VISUAL ANALYTICS (VIS), 2022. **Anais...** [S.l.: s.n.], 2022. p.155–159.

HAN, M. et al. Accelerated Depth Computation for Surface Boxplots with Deep Learning. In: IEEE WORKSHOP ON UNCERTAINTY VISUALIZATION: APPLICATIONS, TECHNIQUES, SOFTWARE, AND DECISION FRAMEWORKS, 2024. **Anais...** [S.l.: s.n.], 2024. p.38–42.

HOCHHEISER, H.; SHNEIDERMAN, B. Dynamic query tools for time series data sets: timebox widgets for interactive exploration. **Information Visualization**, [S.l.], v.3, n.1, p.1–18, 2004.

HYNDMAN, R. J.; SHANG, H. L. Rainbow plots, bagplots, and boxplots for functional data. **Journal of Computational and Graphical Statistics**, [S.l.], v.19, n.1, p.29–45, 2010.

KARNIN, Z.; LANG, K.; LIBERTY, E. Optimal quantile approximation in streams. In: 2016 IEEE 57TH ANNUAL SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE (FOCS). **Anais...** [S.l.: s.n.], 2016. p.71–78.

LAMPE, O. D.; HAUSER, H. Curve density estimates. In: COMPUTER GRAPHICS FORUM. **Anais...** [S.l.: s.n.], 2011. v.30, n.3, p.633–642.

LAMPE, O. D.; HAUSER, H. Interactive visualization of streaming data with kernel density estimation. In: IEEE PACIFIC VISUALIZATION SYMPOSIUM, 2011. **Anais...** [S.l.: s.n.], 2011. p.171–178.

LIMA, P. **Um Estudo Sobre A Computação Aproximada da Profundidade de Banda e sua Aplicação para a Visualização de Grandes Coleções de Séries Temporais.** 2019. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pernambuco.

LIU, D. et al. MTV: visual analytics for detecting, investigating, and annotating anomalies in multivariate time series. **Proceedings of the ACM on Human-Computer Interaction**, [S.l.], v.6, n.CSCW1, p.1–30, 2022.

LÓPEZ-PINTADO, S.; ROMO, J. On the concept of depth for functional data. **Journal of the American Statistical Association**, [S.l.], v.104, n.486, p.718–734, 2009.

MANYIKA, J. Big data: the next frontier for innovation, competition, and productivity. **McKinsey Global Institute**, [S.l.], v.1, 2011.

MORITZ, D.; FISHER, D. Visualizing a Million Time Series with the Density Line Chart. **arXiv preprint arXiv:1808.06019**, [S.l.], 2018.

NARISETTY, N. N.; NAIR, V. N. Extremal depth for functional data and applications. **Journal of the American Statistical Association**, [S.l.], v.111, n.516, p.1705–1714, 2016.

PLAYFAIR, W. **The commercial and political atlas**: representing, by means of stained copper-plate charts, the progress of the commerce, revenues, expenditure and debts of england during the whole of the eighteenth century. [S.l.]: T. Burton, 1801.

PÖTHKOW, K.; WEBER, B.; HEGE, H.-C. Probabilistic marching cubes. In: COMPUTER GRAPHICS FORUM. **Anais...** [S.l.: s.n.], 2011. v.30, n.3, p.931–940.

SUN, Y.; GENTON, M. G. Functional boxplots. **Journal of Computational and Graphical Statistics**, [S.l.], v.20, n.2, p.316–334, 2011.

SUN, Y.; GENTON, M. G.; NYCHKA, D. W. Exact fast computation of band depth for large functional datasets: how quickly can one million curves be ranked? **Stat**, [S.l.], v.1, n.1, p.68–74, 2012.

ZHAO, J. et al. Exploratory analysis of time-series with chronolenses. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.17, n.12, p.2422–2431, 2011.

ZHAO, Y. et al. KD-Box: line-segment-based kd-tree for interactive exploration of large-scale time-series data. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.28, n.1, p.890–900, 2021.

ZHOU, Y. et al. Representation and analysis of time-series data via deep embedding and visual exploration. **Journal of Visualization**, [S.l.], v.26, n.3, p.593–610, 2023.