FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Deteção de falhas em Servidores de Video-on-Demand

João Ponte

Mestrado Integrado em Engenharia Informática e Computação

Orientador: João Pedro Carvalho Leal Mendes Moreira

Co-orientador: Ricardo Santos Morla

21 de Julho de 2014

Deteção de falhas em Servidores de Video-on-Demand

João Ponte

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Hugo Ferreira
Arguente: Carlos Bastos
Vogal: João Mendes Moreira

---

21 de Julho de 2014

Resumo

Os serviços de *Video-on-Demand* permitem aos seus utilizadores ver os vídeos à sua escolha a qualquer hora. Estes serviços têm vindo de certo modo a substituir a televisão dado à liberdade de escolha permitida. Este dinamismo acrescido trás no entanto um aumento de complexidade na estrutura dos seus servidores. Os servidores de *Video-on-Demand* tratam de inúmeros clientes em simultâneo e torna-se difícil detetar anomalias nos dados de tráfego dos servidores com técnicas simples. Este projeto é um estudo à viabilidade de utilização de técnicas de *Data Mining* e processamento de sinal neste problema. Foram aplicados métodos comuns de deteção de *outliers* mas dada a sua fraca prestação em falhas pouco expressivas foi proposto um método para resolver estas situações. Este método recorre extensivamente à distância Dynamic Time Warping (DTW) e parte do pressuposto que os clientes têm comportamentos de um certo modo expectáveis. No final, comparando o método proposto com os métodos mais comuns de deteção de *outliers*, conseguiu-se obter um aumento de $29.5\% \pm 4.3\%$, com 95% de confiança relativamente ao kNN, na deteção de anomalias pouco expressivas.

Abstract

Video-on-Demand services allow users to watch the videos of their choosing at any given time. This kind of services have been replacing television lately due to their freedom of choice. This dynamism however, brings an increase of complexity on the structure of the servers. Video-on-Demand servers handle a lot of simultaneous clients which makes traffic data way too hard to analyse with simple techniques. This project is a viability study of the Data Mining and Signal Processing techniques applied to this problem. Common outlier detection techniques were applied but due to their weak results in finding small faults, a new method was proposed to solve the problem. This method uses the distance of the Dynamic Time Warping (DTW) algorithm extensively and it's premise is that every client has some sort of expectable behaviour. Comparing the proposed method with the most common outlier detection methods, there was a $29.5\% \pm 4.3\%$, with 95% confidence, increase in small anomalies detection comparing to the kNN algorithm.

Conteúdo

1 Introdução e Motivação **1**
- 1.1 Serviços de Video-on-Demand . . . 1
- 1.2 Arquitetura dos Servidores de Video-on-Demand . . . 1
  - 1.2.1 Falhas de tráfego . . . 2
  - 1.2.2 Comportamento dos clientes . . . 2
  - 1.2.3 Comportamento geral do servidor . . . 2
  - 1.2.4 Falhas de tráfego em servidores de Video on Demand . . . 2

2 Deteção de falhas em Séries Temporais **7**
- 2.1 Data Mining . . . 7
- 2.2 Séries Temporais . . . 8
- 2.3 Deteção de Anomalias . . . 8
  - 2.3.1 Outliers . . . 9
  - 2.3.2 Classificação dos métodos de Deteção de Anomalias . . . 11
  - 2.3.3 Classificação dos Dados . . . 11

3 Métodos comuns aplicados a este problema **13**
- 3.1 k-Nearest Neighbors . . . 13
- 3.2 Local Outlier Factor . . . 14
- 3.3 Neural Networks . . . 15

4 Método proposto **17**
- 4.1 Dynamic Time Warping . . . 17
  - 4.1.1 Sobre o DTW . . . 17
  - 4.1.2 DTW em detalhe . . . 17
- 4.2 Distância como Score de anomalias . . . 18
- 4.3 Threshold para a classificação . . . 20

5 Experiências e Resultados **23**
- 5.1 Setting experimental . . . 23
- 5.2 Métodos testados . . . 24
  - 5.2.1 Parâmetros kNN . . . 24
  - 5.2.2 Parâmetros LOF . . . 25
  - 5.2.3 Parâmetros DTW . . . 25
- 5.3 Resultados . . . 26

6 Conclusões **31**
- 6.1 Outros métodos e melhoramentos futuros . . . 31

Referências **33**

A Resultados Comparativos **37**

Lista de Figuras

1.1 Tráfego do cliente em função do tempo. 3
1.2 Tráfego do servidor em função do tempo. 3
1.3 Tráfego do servidor com falhas de 25 em 25 segundos com 1.5 segundos de duração. 4
1.4 Tráfego do servidor com e sem anomalias. 4
1.5 Tráfego do servidor saturado com e sem anomalias. 5

2.1 Outliers Pontuais [CBK09]. 9
2.2 Outliers Coletivos [CBK09]. 10
2.3 Outliers Contextuais [CBK09]. 10

3.1 Classificação kNN com 3 e 5 vizinhos mais próximos. 14
3.2 Classificação kNN com os 2 vizinhos mais próximos. 15
3.3 Exemplo de uma rede neuronal com uma camada escondida [JGP10]. 16

4.1 Comparação de séries usando as medidas de DTW e Euclidiana [FKL+08]. . . . 18
4.2 Esquema do funcionamento do método proposto. 19
4.3 Cruzando os valores obtidos das analises por clientes consegue-se obter informações mais gerais. 20
4.4 Histograma das distâncias com uma distribuição exponencial sobreposta. 21

5.1 Interrupção do tráfego ao longo do tempo. 24
5.2 Exemplo de análise de curvas ROC. 25
5.3 Comparação entre os valores de k a utilizar. 26
5.4 Rácio de verdadeiros positivos para percentagem fixa de falsos positivos. 27
5.5 DTW entre duas séries de clientes, com e sem lower bounding 28
5.6 Resultados com e sem lower bounding. 29

LISTA DE FIGURAS

Lista de Tabelas

5.1 Tempos de execução dos métodos testados 27
5.2 Performance com e sem Lower Bounding 28

LISTA DE TABELAS

Abreviaturas e Símbolos

DNS	Domain Name System
DTW	Dynamic Time Warp
kNN	k-Nearest Neighbour
LOF	Local Outlier Factor
ROC	Receiver Operating Characteristic

Capítulo 1

Introdução e Motivação

Neste capítulo será feita uma introdução ao objetivo do trabalho assim como uma análise aos servidores de *Video-on-Demand*.

1.1 Serviços de Video-on-Demand

O *Video-on-Demand* é um serviço que permite aos utilizadores assistir a qualquer vídeo à sua escolha, em contraste com os serviços de televisão por exemplo, onde a programação é estática e o utilizador não escolhe qual vídeo quer ver. Este dinamismo associado aos serviços *Video-on-Demand*, faz com que estes precisem de estruturas adequadas, já que cada utilizador pode escolher o vídeo que quiser ver e quando quiser ver. Falhas, independentemente do tipo, vão acontecer numa estrutura de alta escala, como o caso das destes serviços. A sua deteção para posterior resolução, deve ser tida em conta para que o serviço seja viável. Uma vez que os servidores destes serviços podem lidar com vários clientes, a quantidade de informação transmitida pode ser demasiado alta para ser analisada sem usar técnicas que consigam lidar com muitos dados. É portanto uma ideia a considerar, a utilização de técnicas de *Data Mining* e porventura de Processamento de Sinal para resolver este problema.

1.2 Arquitetura dos Servidores de Video-on-Demand

As arquiteturas de serviços de *Video-on-Demand* podem variar mas o recurso a múltiplos servidores quer seja para efeitos de aumento de tráfego disponível, redundância de dados, *cache*, melhoria de serviço em determinados locais, ou simplesmente aumento de espaço disponível, é quase obrigatório em serviços de larga escala como é o caso do YouTube. O estudo do modelo destes serviços é importante para o a contextualização do problema. Para ter uma ideia de como estes normalmente se estruturam, foi realizada uma pesquisa usando o YouTube como referência,

já que este é o serviço mais popular de *Video-on-Demand*. De acordo com o estudo feito em [AJClZ11], o YouTube é composto por três componentes distintas:

- **Espaço de identificadores dos vídeos linear** — os identificadores dos vídeos são atribuídos sem qualquer tipo de hierarquia;
- **Cache composta por 3 níveis** — cada nível corresponde à prioridade que os vídeos têm, vídeos mais populares encontram-se nas *cache* de nível mais baixo;
- **Múltiplos namespaces de DNS (*Domain Name System*)** — que formam uma hierarquia de servidores.

A lógica principal da organização do YouTube centra-se na delegação da redistribuição conforme a hierarquia suposta para os servidores de DNS, permitindo assim a criação dos espaços de identificadores lineares que por si permite a utilização de *hash* fixa para um rápido acesso ao armazenamento dos vídeos.

1.2.1 Falhas de tráfego

As falhas num servidor *Video-on-Demand* podem ter diversas origens, quer seja numa má ligação física ou falha de *software*, mas a maneira como se manifestam é geralmente na forma de falhas no tráfego. A perda de ligação, atraso de entrega de pacotes ou problemas de *routing*, têm todos efeitos na qualidade final de tráfego. É portanto necessário antes de descobrir quais as origens dos problemas, descobrir se há problemas, quando e com que frequência estes acontecem.

1.2.2 Comportamento dos clientes

O comportamento dos clientes do YouTube inicia-se com um *pre-buffering* de cerca 55 segundos do início do vídeo. De seguida, enquanto o vídeo é reproduzido, o tamanho do *buffer* reduz mas tenta manter um nível constante pedindo blocos mais pequenos constantemente [JWH+13]. O simulador utilizado para os testes reproduziu este comportamento que pode ser visto na figura 1.1.

1.2.3 Comportamento geral do servidor

O tráfego do servidor é a soma do tráfego dos seus clientes. Uma vez que estes podem estar em fases diferentes do *buffering* (55 segundos iniciais, a pedir pacotes e entre pedidos), o comportamento do servidor é caótico e de difícil análise direta. Na figura 1.2 vê-se a representação gráfica de uma parte do tráfego do servidor simulado.

1.2.4 Falhas de tráfego em servidores de Video on Demand

Falhas momentâneas de tráfego numa situação como a da figura 1.2, podem ser facilmente detetadas como sendo descidas bruscas caso estas durem tempo suficiente. Na figura 1.3 são

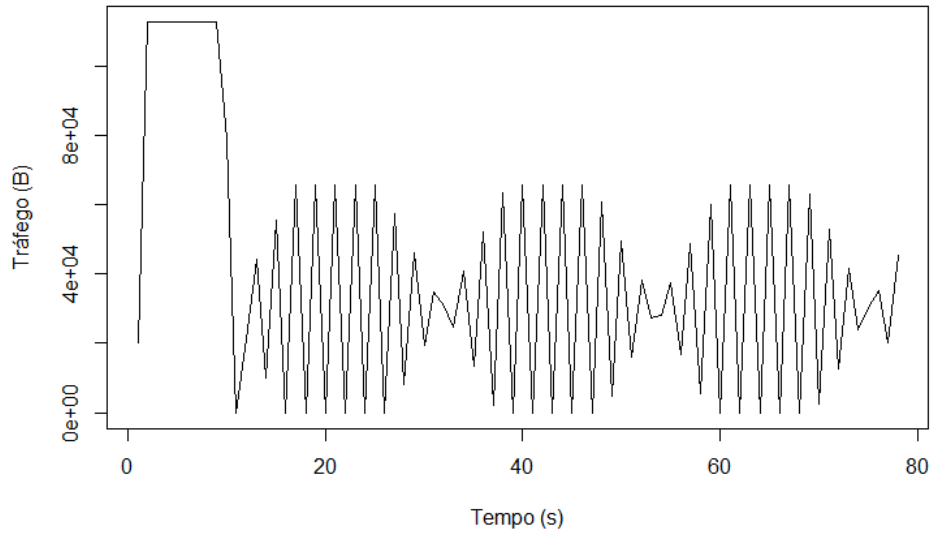


Figura 1.1: Tráfego do cliente em função do tempo.

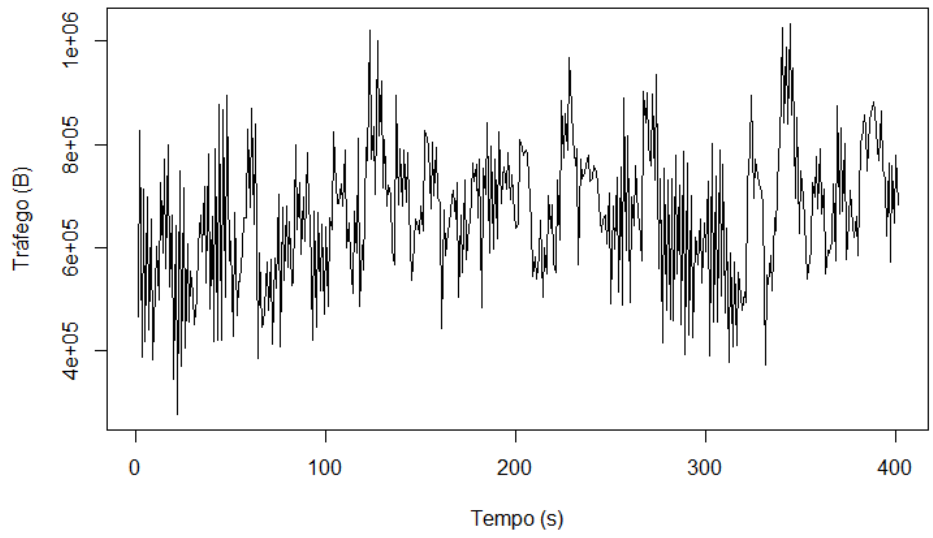


Figura 1.2: Tráfego do servidor em função do tempo.

notórias as falhas de 1.5 segundos de duração. No entanto, quando estas duram pouco tempo, a deteção torna-se mais difícil. Na figura 1.4 vê-se um exemplo de tráfego normal simulado e um onde de 5 em 5 segundos ocorre uma pequena falha de tráfego de 0.1 segundos. Estas falhas são demasiado pequenas para se notar nestas situações.

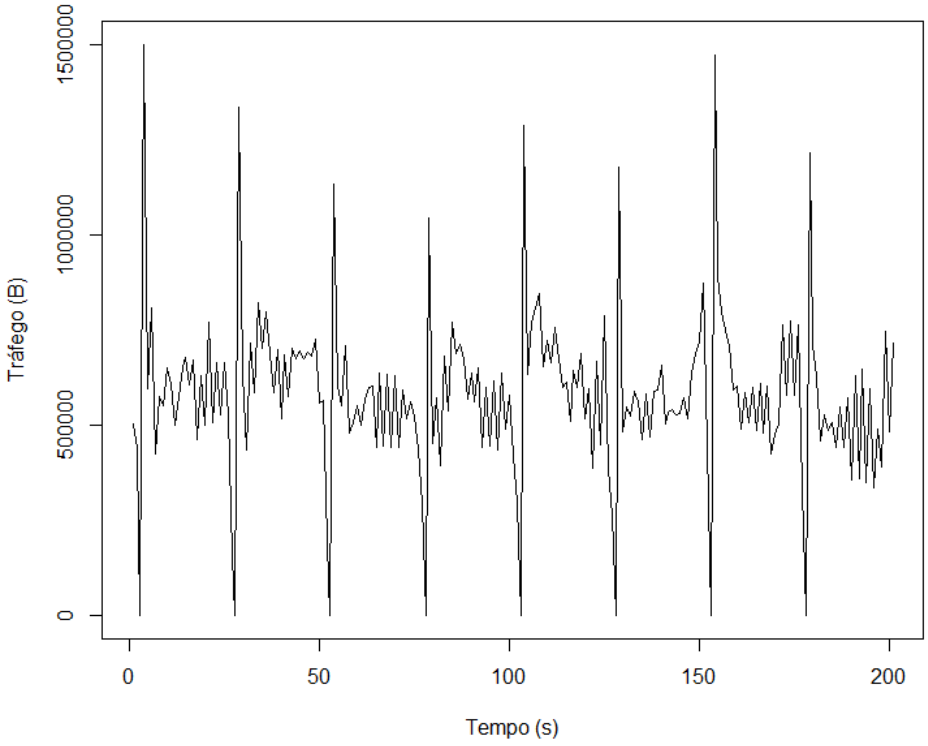
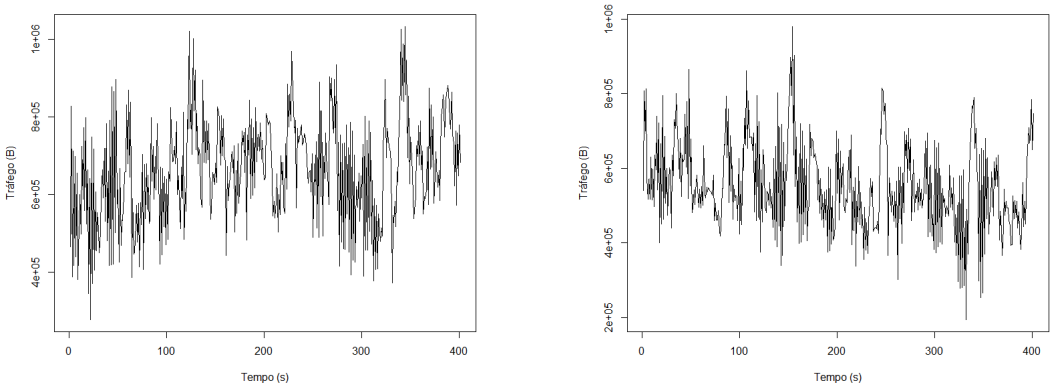


Figura 1.3: Tráfego do servidor com falhas de 25 em 25 segundos com 1.5 segundos de duração.



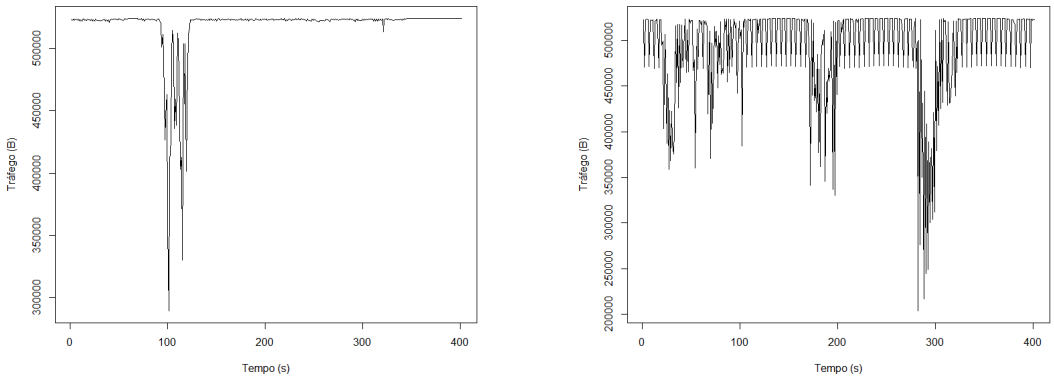
(a) Tráfego Normal.

(b) Tráfego com anomalias.

Figura 1.4: Tráfego do servidor com e sem anomalias.

Outro cenário possível que exemplifica a influência do contexto na deteção das falhas é o de um servidor com saturação de tráfego. Num caso saturado, os pedidos de dados dos clientes são demasiado elevados para a capacidade do servidor. Nesta situação, o servidor fica com o tráfego atual constantemente no limite da sua capacidade com ocasionais descidas devido à saída de cli-

entes. Isto faz com que o servidor tenha um comportamento muito mais regular que o anterior. A figura 1.5 representa um servidor saturado com as mesmas anomalias do anterior. Ambas as situações são simuladas.



(a) Tráfego Normal.

(b) Tráfego com anomalias.

Figura 1.5: Tráfego do servidor saturado com e sem anomalias.

Capítulo 2

Deteção de falhas em Séries Temporais

Neste capítulo, será abordado o estado da arte da deteção de falhas em séries temporais assim como uma introdução acerca do *Data Mining*.

2.1 Data Mining

O *Data Mining* é a análise de grandes quantidades de dados de modo a obter informação relevante. Esta informação é obtida através da procura e identificação de padrões nos dados. Técnicas de *Data Mining* são utilizadas constantemente em inúmeras situações que vão desde processos industriais, análises de texto, imagem e ciências biomédicas. Cabe depois ao analista que utiliza estas técnicas, decidir quais os padrões realmente interessantes ou não [Run12, HTF01]. Segundo [She00], o *Data Mining* pode-se dividir em 5 fases por ordem:

1. **Buisness Understanding** — Fase onde se determinam quais os objetivos da análise;
2. **Data Understanding** — Coleção inicial dos dados e descrição;
3. **Data Preparation** — Fase de seleção dos dados relevantes e formatação caso necessária;
4. **Modeling** — Seleção e construção de uma técnica de modelação;
5. **Evaluation** — Avaliação dos resultados obtidos;
6. **Deployment** — Implementação do plano construido e/ou produção de um relatório final.

Esta organização é utilizada genericamente em todos os projetos de *Data Mining* independentemente da sua complexidade. Em relação às ferramentas, existe neste momento uma grande quantidade de software disponível [WWG11]. Embora todos eles tenham capacidade de realizar as funções mais básicas do *Data Mining*, a metodologia de cada um e o seu foco variam. As ferramentas analisadas para a realização deste projeto foram:

- **R** — Uma das linguagens estatísticas mais populares, tem uma grande base de bibliotecas com métodos de *Data Mining* disponível, facilidade em criar gráficos de alta qualidade e uma comunidade de utilizadores ativa;
- **Rapid Miner** — Uma ferramenta mais recente que o R, com um sistema simples e limpo de drag and drop que permite uma rápida e fácil prototipagem dos métodos;
- **Matlab** — Ferramenta de computação matemática que contém inúmeras funções quer de *Data Mining*, de processamento de sinal e de *Machine Learning* e conta com uma documentação extremamente detalhada.

No início do projeto foram abordadas as três ferramentas embora a escolha final tenha recaído no R, principalmente devido à extensão e qualidade das bibliotecas necessárias para o trabalho.

2.2 Séries Temporais

Séries temporais são sequências de dados intervalados usualmente de forma uniforme, embora haja casos onde os intervalos são diferentes entre si. Em caso de interrupções, como um fecho da bolsa por exemplo, é habitual que se separem as séries temporais. As mais comuns são unidimensionais mas podem ser também multidimensionais, isto é, haver dois ou mais atributos a acontecer ao mesmo tempo. Este tipo de dados traz restrições na sua análise devido à correlação que os atributos têm com o tempo. As séries temporais são comuns em análises financeiras para mostrar as progressões dos mercados mas são usadas em muitas outras áreas. O tráfego de um servidor, caso em estudo neste trabalho, pode ser representado numa série temporal onde a cada intervalo de tempo corresponde a quantidade de dados transmitida como foi visto na figura 1.2.

As abordagens feitas às séries temporais, de uma maneira geral, são feitas analisando os dados através dos seguintes pressupostos [SS06]:

- **Tempo** — Neste caso, considera-se que existe correlação temporal entre os valores, isto é, todos os eventos da série, dependem de certo modo dos valores que os antecedem;
- **Frequência** — Nesta abordagem, parte-se do pressuposto que existem características periódicas nas séries e são essas características as de maior relevância no estudo.

As séries temporais podem ser abordadas geralmente através de duas maneiras, pelo domínio da frequência e pelo domínio do tempo. A abordagem pelo domínio do tempo parte do pressuposto que existe uma dependência entre o valor atual e os passados de modo a que haja uma correlação entre valores adjacentes. Já a segunda abordagem tem como foco as características periódicas das séries .

2.3 Deteção de Anomalias

O estudo da deteção de anomalias, vulgarmente denominada também como deteção de outliers, já é feito por estatísticos desde o século XIX [Edg87] e tem vindo a ser adotado nas mais

diversas áreas, como a deteção de fraudes [BHH01], de falhas em sistemas críticos [HA02] ou de ataques em sistemas informáticos [ASG04]. A deteção de anomalias já foi apresentada várias vezes aplicada a redes embora no contexto de deteção de intrusões. Nestas situações, os sistemas utilizados tentam encontrar padrões no tipo de tráfego da rede que não correspondem ao esperado ou que correspondem a ataques conhecidos [BBK14]. Estes métodos diferem do caso em estudo no tipo de dados em análise. Enquanto que na deteção de intrusões é analisado o conteúdo e os tipos de pacotes transmitidos, no caso em estudo é só analisado a quantidade de tráfego a cada momento. No que se refere às anomalias em concreto, estas podem ter diversos tipos e ser classificadas de diferentes maneiras. Os métodos de deteção devem ter em vista o tipo de anomalia que querem procurar. Nas secções seguintes são explorados os tipos de anomalias (*outliers*) existentes assim como os diferentes métodos de deteção e a maneira como estes os classificam.

2.3.1 Outliers

Outliers são observações que não agem conforme o comportamento considerado normal de um conjunto de dados. As suas origens variam conforme o objeto de estudo e podem significar, entre outros problemas, falhas em instrumentos de observação, operações maliciosas ou até problemas médicos [CBK09]. Os *outliers* são geralmente divididos em três categorias: Pontuais, Contextuais e Coletivos. De seguida encontra-se uma descrição mais detalhada de cada um deles.

- **Pontuais** — Um *outlier* pontual é um evento que é considerado anómalo porque se "distancia" dos outros eventos. A noção de distância pode variar conforme os dados mas a ideia geral é de que os valores do evento são demasiado diferentes de todos os outros ou pelo menos da maioria dos eventos classificados como normais. A figura 2.1 representa graficamente algumas anomalias pontuais. Os pontos O1 e O2 e o conjunto O3 encontram-se distantes dos conjuntos N1 e N2 considerados normais e são por isso classificados como outliers.

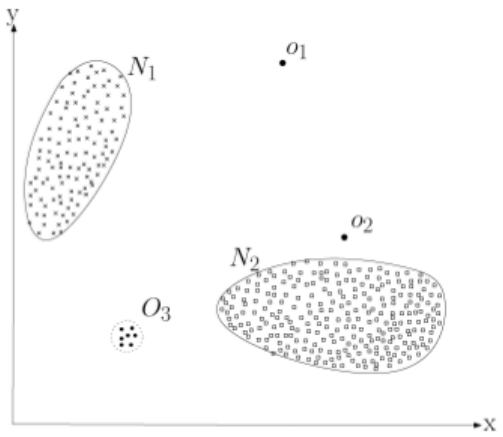


Figura 2.1: Outliers Pontuais [CBK09].

- **Coletivos** — Estes outliers são vários eventos que, quando ocorridos isoladamente ou em determinados contextos, são normais, mas quando ocorridos num determinado conjunto são considerados anomalias. A figura 2.2 representa um eletrocardiograma onde é possível observar uma anomalia correspondente a uma contração arterial prematura. Os valores da anomalia em si não são anormais, mas uma vez que se repetem durante algum tempo criam uma anomalia.

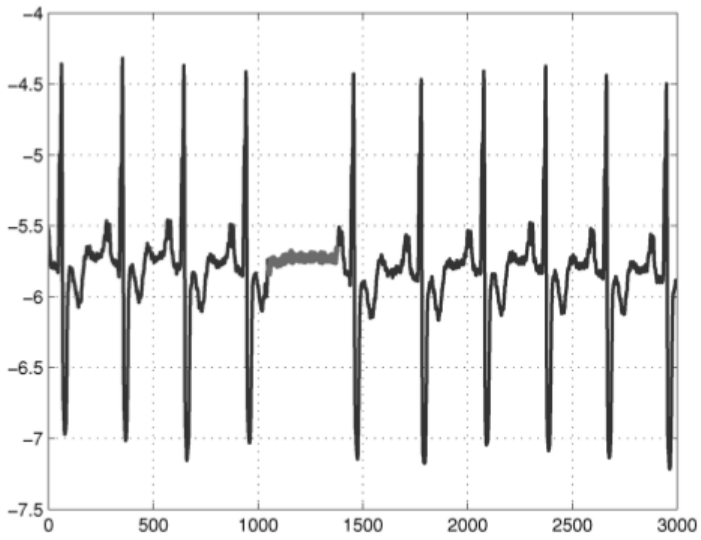


Figura 2.2: Outliers Coletivos [CBK09].

- **Contextuais** — Neste caso, o evento pode ter valores que são normais em relação aos dados restantes, mas em relação aos eventos vizinhos (ao contexto), é uma anomalia. A figura 2.3 representa uma série temporal com a temperatura ao longo do ano. A temperatura *t2* é uma anomalia, não pelo seu valor ser anormal (é igual ao de *t1* que não é anomalia) mas pelo facto de esse valor não ser normal naquela altura do ano.

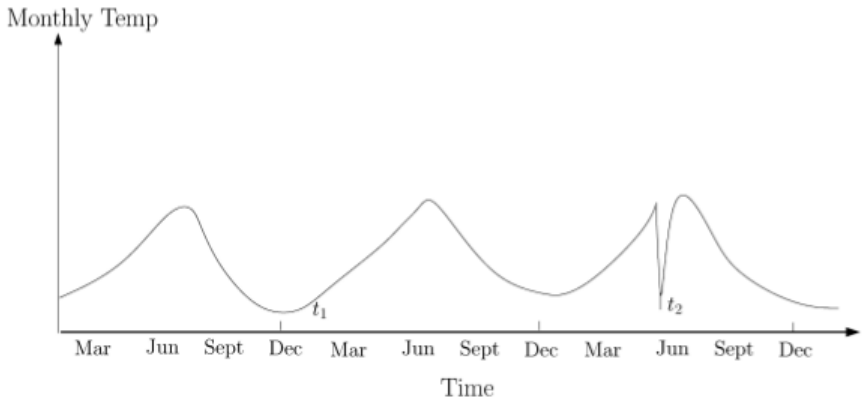


Figura 2.3: Outliers Contextuais [CBK09].

2.3.2 Classificação dos métodos de Deteção de Anomalias

Os métodos de deteção de anomalias podem ser enquadrados em diferentes categorias conforme a maneira como classificam os dados. As principais categorias destes algoritmos são, segundo [CBK09], os seguintes:

- **Supervisionado** — Os métodos supervisionados requerem dados de treino previamente marcados. Dados de treino são conjuntos de dados exemplo que servem para definir tanto os padrões expectáveis como as anomalias. Os eventos a ser analisados são então comparados com os dados de treino e classificados de acordo;
- **Não Supervisionado** — Estes métodos não requerem dados de treino. Simplesmente assumem que os eventos normais são mais frequentes que as anomalias. Estes métodos são versáteis mas em casos em que as anomalias são frequentes estes podem não produzir os resultados esperados;
- **Semi-Supervisionado** — Estes métodos utilizam dados de treino mas apenas os eventos normais são classificados. Estes métodos consideram então que todos os eventos que se desviam demasiado dos dados de treino são anomalias.

2.3.3 Classificação dos Dados

Os valores retornados pelos métodos de deteção de outliers são as classificações dos eventos como sendo anomalias ou não. Existem duas maneiras distintas de classificação dos dados: *Labeling* e *Scoring*. A diferença entre as duas é essencialmente a sua assertividade perante a classificação de um evento como sendo *outlier* ou não. A utilização destes tipos de classificação varia essencialmente conforme o método utilizado. As classificações de acordo com [CBK09] são:

- **Labeling** — Classificação binária. O evento é simplesmente classificado como anómalo ou normal. Um exemplo de método que normalmente classifica usando *labeling* é o de redes neuronais.
- **Scoring** — Ao evento é associado um valor que quantifica a possibilidade de este ser ou não um *outlier*. Métodos que utilizam esta classificação são por exemplo os estatísticos, em que ao evento é associada uma probabilidade de este ser ou não *outlier*.

Capítulo 3

Métodos comuns aplicados a este problema

O ponto de partida para a análise do problema foi analisar e testar os métodos mais comuns de detecção de anomalias. A escolha dos algoritmos baseou-se principalmente na sua capacidade de se adaptarem a séries temporais e de detetarem anomalias pontuais ou contextuais. Numa fase inicial foi utilizado o software *RapidMiner*, que permite uma rápida prototipagem e análise geral do funcionamento dos métodos. O procedimento passava por analisar os dados simulados tanto dos clientes como do servidor. Nesta fase, os métodos não revelaram resultados muito favoráveis em anomalias pouco evidentes, daí a necessidade de criar um método mais específico para o contexto do problema. Em seguida será feita uma análise de três métodos e como foram aplicados.

3.1 k-Nearest Neighbors

O kNN é um método de classificação e regressão relativamente simples. A premissa principal do método é a de que dados semelhantes se agregam entre si, logo a distancia entre eles é baixa. No caso da detecção de anomalias é esta distância aos seus vizinhos mais próximos que define um evento de anomalia ou não [CBK09]. O funcionamento básico do kNN, descrito em [BR98], é o seguinte: para cada evento o seu *score* de anomalia é igual à distância do seu k vizinho mais próximo. A distância usada normalmente é a euclidiana, no entanto podem ser utilizadas outras assim como pode ser utilizada a média das distâncias dos k elementos. Existem inúmeras variações desta técnica, aplicadas consoante o problema [CBK09]. Este método pode tanto ser como não ser supervisionado. No caso do supervisionado, este calcula a distância dos eventos a classificar aos eventos já classificados; no caso do não supervisionado, ele calcula as distâncias entre todos os eventos. A figura 3.1 mostra a classificação de um elemento que pode ser classificado de duas maneiras conforme o número de vizinhos utilizados.

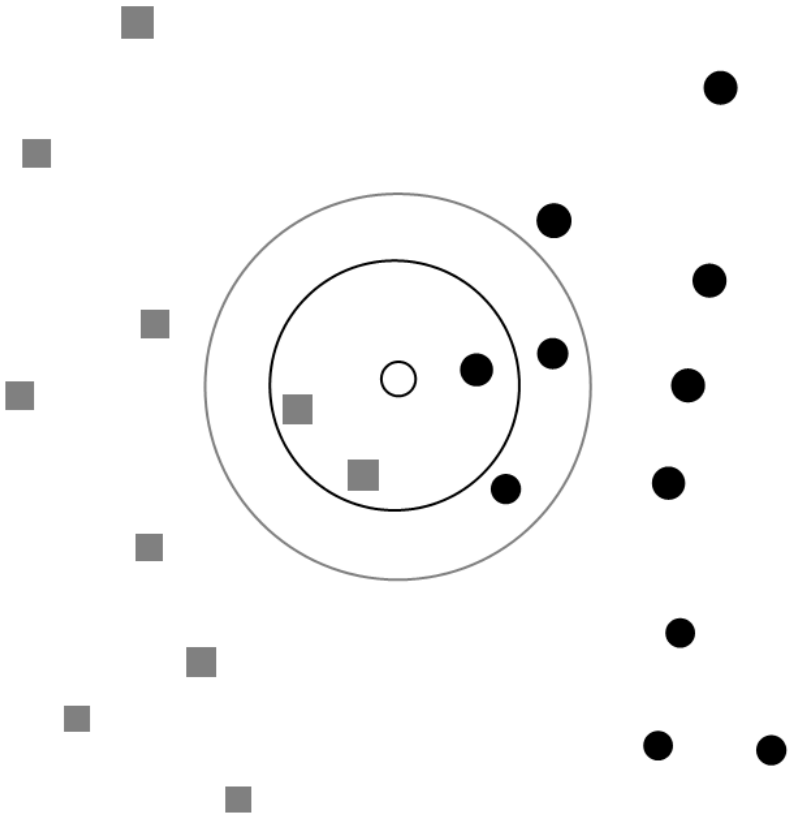


Figura 3.1: Classificação kNN com 3 e 5 vizinhos mais próximos.

Para séries temporais o kNN utiliza como vizinhos mais próximos, os eventos mais próximos no tempo, ou seja, imediatamente antes e depois do evento alvo de classificação, e usa a diferença de amplitudes como distância entre os atributos correspondentes como pode ser visto na figura 3.2.

Através de testes iniciais no *RapidMiner*, deu para perceber que o kNN teria algum potencial em encontrar anomalias que tivessem alguma expressividade na série.

3.2 Local Outlier Factor

O LOF pode ser considerado como uma variação do kNN. Este parte do mesmo princípo de proximidade, mas utiliza o rácio da sua densidade e das dos k vizinhos mais próximos. A densidade de uma observação é calculada através da relação entre k e o volume do espaço circundante dessa mesma observação, usando como raio a distância ao k vizinho mais próximo[BKNS00]. Este método foi aplicado de modo semelhante ao do kNN para séries temporais, onde os vizinhos

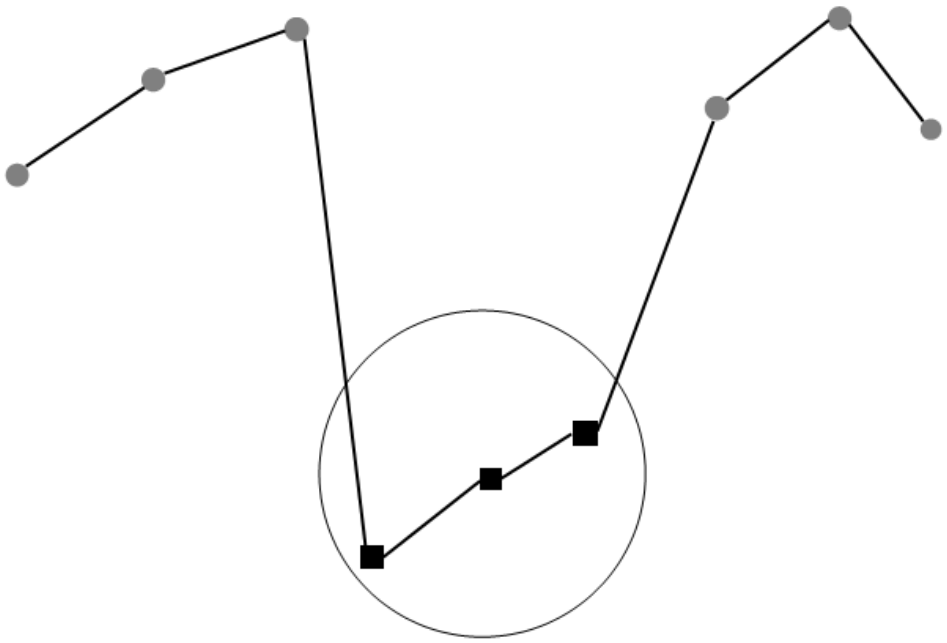


Figura 3.2: Classificação kNN com os 2 vizinhos mais próximos.

são temporais e a medida de distância é a diferença de amplitudes. Uma vez que o funcionamento deste método é semelhante ao kNN, foi pressuposto que os resultados fossem parecidos ou até melhores, daí a sua inclusão para os testes seguintes.

3.3 Neural Networks

As redes neuronais são sistemas de aprendizagem inspirados nos sistemas nervosos encontrados no sistema nervoso do nosso cérebro. Estes modelos consistem em redes de neurónios, distribuídas por várias camadas que ajustam os valores das ligações entre si de acordo com os valores de entrada de treino. Dessas camadas uma é de entrada, outra de saída e as restantes são denominadas de camadas escondidas. A figura 3.3 mostra um exemplo de uma rede neuronal.

As redes neuronais foram usadas extensivamente para a deteção de anomalias como referido em [MS03]. Estes métodos provaram ser eficientes na classificação de dados mas para uso em

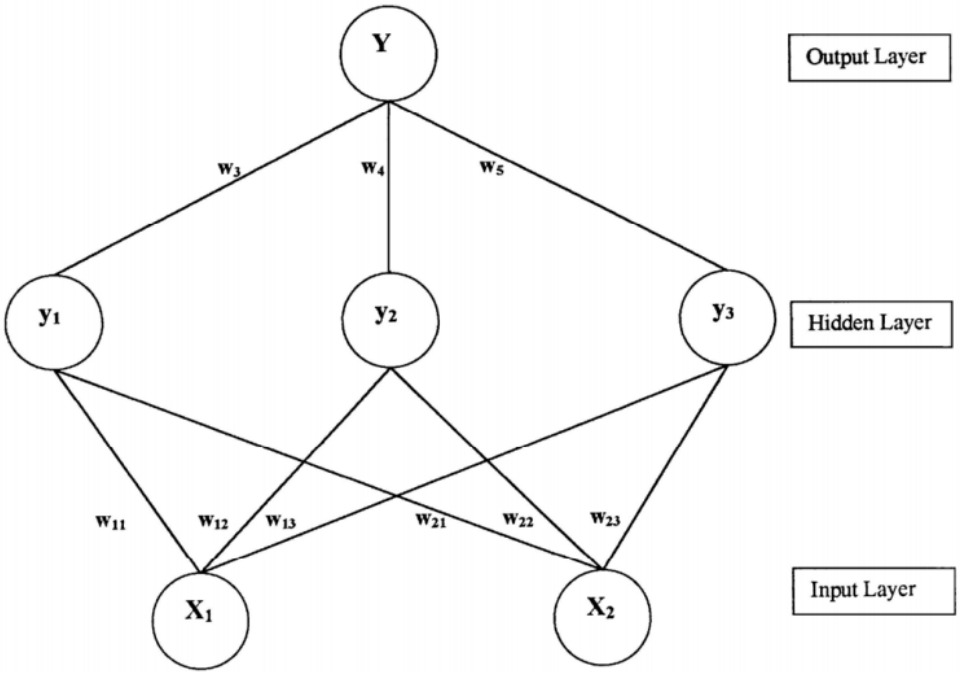


Figura 3.3: Exemplo de uma rede neuronal com uma camada escondida [JGP10].

séries temporais a investigação é mais escassa. Em [KM13], redes neuronais são usadas para previsão do comportamento de séries de mercados assim como em [JGP10] foram utilizadas para prever consumo de energia. Os resultados foram favoráveis mas ambos os casos têm como pressuposto séries onde existem tendências bem definidas ou acontecimentos periódicos, algo que não se encontra ao nível do caso em estudo. Este tipo de métodos são uteis para séries de longos períodos de tempo, onde se formam tendências ou componentes sazonais. Alguns testes no *RapidMiner* permitiram também perceber que a aprendizagem deste algoritmo não tinha precisão suficiente para encontrar anomalias, daí ter sido excluído dos testes finais.

Capítulo 4

Método proposto

Para resolver o problema foi criado um método de análise das séries dos clientes. Este método usa extensivamente o algoritmo de Dynamic Time Warping (DTW) que será explicado na secção seguinte. A escolha do DTW para comparar séries vem do facto de este ser muito útil quando as séries não variam na forma mas têm durações diferentes. Neste caso, o comportamento da reprodução de vídeo é regular mas diferenças de largura de banda podem alterar a duração das fases de *buffering*.

4.1 Dynamic Time Warping

4.1.1 Sobre o DTW

O DTW é um algoritmo usado para comparar duas séries temporais. A vantagem do DTW é que este consegue esticar ou comprimir localmente as séries de modo a conseguir alinhá-la as duas séries. Isto é vantajoso em séries que são semelhantes na forma mas decorreram a velocidades diferentes [FKL+08]. No figura 4.1 vê-se a comparação entre duas séries semelhantes mas em que uma delas decorreu num espaço de tempo mais curto que a outra.

A distância euclidiana, a mais comummente utilizada [LKLC03], não consegue identificar esta semelhança enquanto que o DTW consegue esticar a segunda série de modo a alinhar com a primeira. O DTW tem sido extensivamente utilizado para diversas utilizações principalmente para reconhecimento de voz. [MGB12, CDLDS13, PG08].

4.1.2 DTW em detalhe

A descrição e símbolos seguintes foram baseados no *Computing and Visualizing Dynamic Time Warping Alignments in R* [Gio09], visto este ter a descrição da biblioteca em R utilizada no *setup* experimental. Começa-se por definir como *query* e *template* as séries a comparar assim

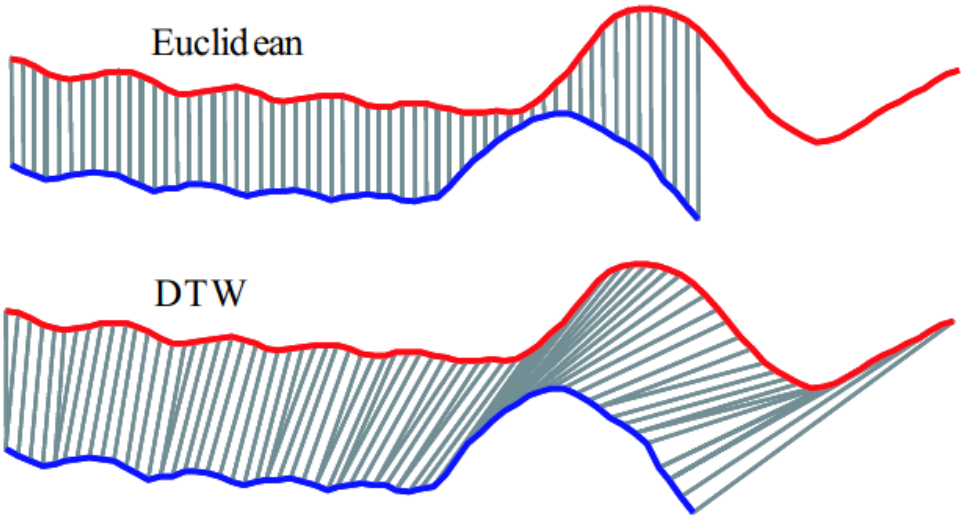


Figura 4.1: Comparação de séries usando as medidas de DTW e Euclidiana [FKL+08].

como uma função de distância *d(i,j)*, sendo *i* e *j* índices das séries. *Query* e *template* têm como comprimento *N* e *M* respectivamente.

$$query : X = (x_1, \ldots, x_N) \tag{4.1}$$

$$template : Y = (y_1, \ldots, y_M) \tag{4.2}$$

$$d(i, j) = f(x_i, y_j) \tag{4.3}$$

O principal componente do DTW é a *warping curve* ϕ, que deforma temporalmente as séries alterando os índices iniciais. Considera-se então como ϕ_x e ϕ_y as *warping curves* de x e y respetivamente, assim como $d_\phi(X,Y)$ a distância entre as *warping curves* das séries X e Y.

$$\phi_x(k) \in \{1 \ldots N\}, \tag{4.4}$$

$$\phi_y(k) \in \{1 \ldots M\} \tag{4.5}$$

A ideia do DTW é então encontrar a deformação de modo a alinhar as séries e minimizar a distância total entre elas, ou seja, minimizar $d_\phi(X,Y)$.

4.2 Distância como Score de anomalias

A ideia inicial consistia em utilizar o DTW comparando as séries a analisar com uma base de séries tidas como normais. Uma vez que o DTW retorna a distância entre as séries *query* e *template*, essa distância pode ser utilizada como o *score* da anomalia. Sendo *template* uma série considerada normal, quanto maior a distância entre as duas mais probabilidade há de haver uma anomalia na série *query*. O DTW poderia ser portanto aplicado tanto à série de tráfego total como

à de cada cliente. Inicialmente o método foi aplicado ao tráfego total com a expetativa de encontrar a longo prazo, pequenos padrões no tráfego que pudessem ser utilizados para comparar. Em [MGB12] é descrito um modelo não supervisionado para aquisição de *motifs* (padrões que não ocorrem necessariamente de forma periódica) em séries de áudio. O método de divisão das séries utilizado para comparação é o de deteção de silêncios [PG08], uma vez que se trata de reconhecimento de voz. Neste caso o método de segmentação terá de ser diferente, no entanto o modelo é de todo semelhante. A segmentação das séries ganhou importância quando se começou a fazer *Data Mining* em grandes séries e o armazenamento requeria poupança de espaço ou indexação eficiente como descrito em [KCHP93]. Nesse mesmo artigo são apresentadas três categorias de algoritmos para a segmentação de séries:

- **Sliding Window** — o tamanho do segmento cresce até que um certo critério de erro seja excedido e passa para o próximo segmento;
- **Top Down** — a série é segmentada recursivamente até ser cumprido um determinado critério de paragem;
- **Bottom Up** — a partir da melhor aproximação possível, ir juntando segmentos até o critério de paragem ser cumprido.

Uma vez que o método teria como objectivo a detecção de outliers em tempo real, o *Sliding Window* seria o algoritmo mais adequado uma vez que pode ser utilizado. O critério utilizado foi o da variação brusca da média dos 3 últimos eventos analisados da janela. A figura 4.2 representa o funcionamento do método.

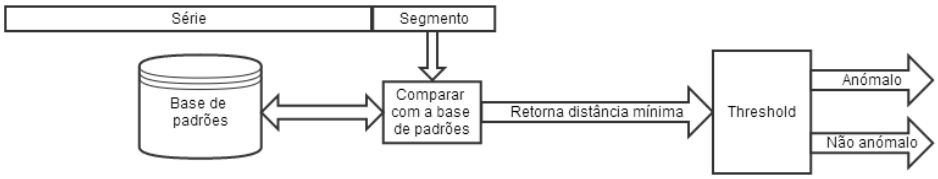


Figura 4.2: Esquema do funcionamento do método proposto.

No entanto, séries de cerca de 7000 eventos não revelaram existência de padrões suficientemente fortes para que funcionasse. Além disso, sem qualquer tipo de indexação dos segmentos guardados, o método tem enormes problemas de performance. Optou-se então, que o mesmo método fosse utilizado para analisar o tráfego de cada cliente já que estes têm comportamentos semelhantes e onde os padrões são mais óbvios. Existe então uma fase inicial onde são guardadas séries de clientes com comportamento considerado normal que será utilizada depois como referência. A classificação de dados como sendo normais deve partir de quem implementa o algoritmo. Estes podem vir de simulações ou de execuções onde haja certeza que não há falhas. Esta base de dados de clientes normais deve conter séries o mais diversas possíveis mas também o mais distintas, já que séries muito idênticas só irá trazer redundância e aumento do tempo de pesquisa. Uma

vez que os tipos de padrões são poucos nas séries dos clientes não é necessária a indexação dos segmentos. Depois, à chegada de cada cliente, é feito o cálculo da distância dos novos segmentos com os da base de dados. É depois utilizada a distância mínima obtida como *score* de anomalia, quanto maior a distância, maior é a probabilidade de haver uma anomalia naquele segmento. Finalmente, uma vez que se tem informação do tempo de chegada de todos os clientes, os *scores* de cada cliente podem ser agregados. Isto permite determinar se a anomalia ocorreu em todos ou num cliente só, eliminar falsos alarmes ou até descobrir com mais detalhe do momento em que ocorreu a anomalia. Este processo é descrito no esquema 4.3, onde se vêm quatro clientes divididos em segmentos, neste caso de tamanho fixo, sobrepostos de acordo com a ordem de chegada. As linhas verticais indicam que segmentos são utilizados para calcular o *score* de anomalia naquele ponto.

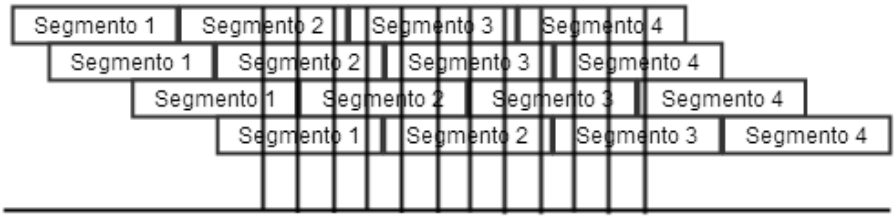


Figura 4.3: Cruzando os valores obtidos das analises por clientes consegue-se obter informações mais gerais.

4.3 Threshold para a classificação

Como foi referido anteriormente, o DTW retorna a distância entre séries que é utilizada como *score* de anomalia. Mas para que haja classificação de um evento como anómalo ou não, é preciso que haja um *threshold* a partir do qual se classifica de anómalo ou não um evento. Este *threshold* pode ser um valor fixo, obtido através de experiências anteriores que tivesse optimizado os resultados. A variação da eficácia de um threshold pode ser vista numa curva de *Receiver operating characteristic*. No ROC é possível ver a relação entre os rácios falsos e verdadeiros positivos fazendo variar o *threshold*. Esta curva tem como principal objectivo ver a eficácia de um método de classificação é descrita com mais detalhe na secção 5.3. Utilizando um *threshold* diretamente na distância é considerar que a distância é já um score por si mesma. Uma maneira de transformar a distância num valor de probabilidade com mais significado seria a seguinte: calcular as distâncias mínimas entre clientes normais e a base de dados e utilizar esses dados para modelar uma distribuição estatística. Para descobrir que tipo de distribuição melhor se adapta ao problema foi criado um histograma com todas as distâncias mínimas entre clientes obtidos pelo servidor simulado.

Na figura 4.4 vê-se o histograma dos mínimos calculados e uma distribuição exponencial sobreposta utilizando o inverso da média desses mesmos mínimos. O único parâmetro da distribuição exponencial, o λ, pode ser calculado pelo inverso da média. Com esta distribuição, podemos

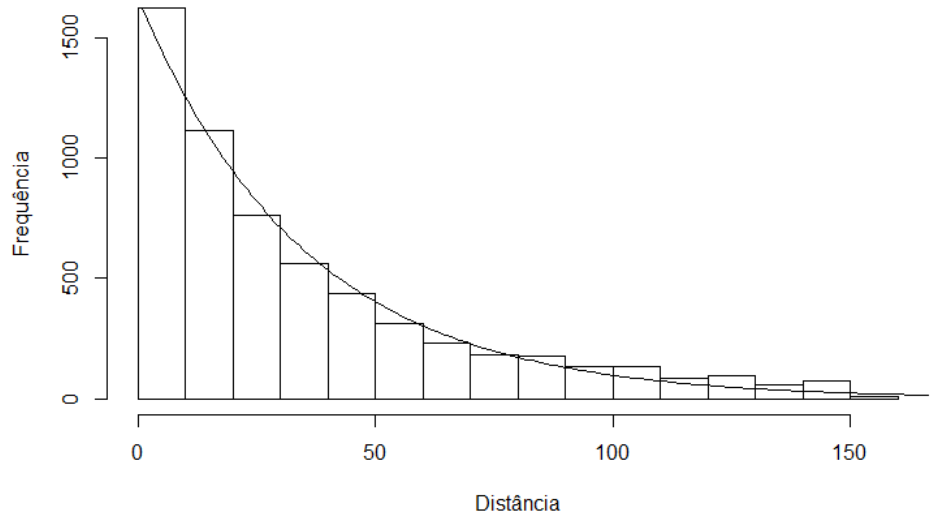


Figura 4.4: Histograma das distâncias com uma distribuição exponencial sobreposta.

então a cada distância retornada pelo método, verificar onde é que esta se encontra na curva de probabilidade da distribuição e conseguir assim obter uma probabilidade de anomalia. De notar que este histograma foi obtido a partir de um servidor simulado com as características descritas na secção 1.2.2. Para problemas onde funcionamento dos clientes difere, deve ser gerado uma distribuição com o funcionamento do mesmo.

Capítulo 5

Experiências e Resultados

5.1 Setting experimental

Todos os testes foram realizados em R, uma linguagem focada em estatística que tem vários métodos de *Data Mining* disponíveis assim como excelentes ferramentas de visualização. Os dados do servidor foram obtidos por um simulador em Java fornecido pelo Co-Orientador. O código do servidor encontra-se nos documentos da dissertação. Este retorna séries com o tráfego total, por cliente e o número de clientes ativos. O simulador comporta-se à semelhança do comportamento descrito na secção 1.2.2. Os resultados retornados são amostras ao segundo, no entanto o servidor simula a uma frequência mais alta. Isto faz com que as falhas com menos de um segundo sejam menos explícitas já que a cada intervalo de um segundo é mostrado o total de tráfego nesse intervalo. Embora intuitivamente uma amostragem mais precisa ou com mais frequência pudesse facilitar a deteção, é pouco viável porque ia gerar uma quantidade de dados insuportável e puderia haver um *overhead* demasiado grande na coleção dos dados. São gerados pedidos ao servidor de clientes com diferentes larguras máximas de banda, intervalados aleatoriamente entre 0 e 10 segundos. Para testar os métodos, o simulador retornou simulações com interrupções em intervalos e tempos regulares. Para comparar a precisão dos métodos, o tempo das interrupções variou entre 0.1 segundos e 2 segundos com intervalos de 25 segundos entre si (figura 5.1).

Para a análise dos resultados de métodos de classificação, é comum recorrer-se a curvas ROC [MS03]. Numa curva ROC, quanto mais a curva se aproxima do ponto *(0,1)*, ou seja, quanto maior o rácio de verdadeiros positivos e menor o de falsos positivos, mais eficiente é o método. Estas curvas conseguem mostrar com clareza quão bom é um método de classificação. A figura 5.2 é um simples guia de análise destas curvas.

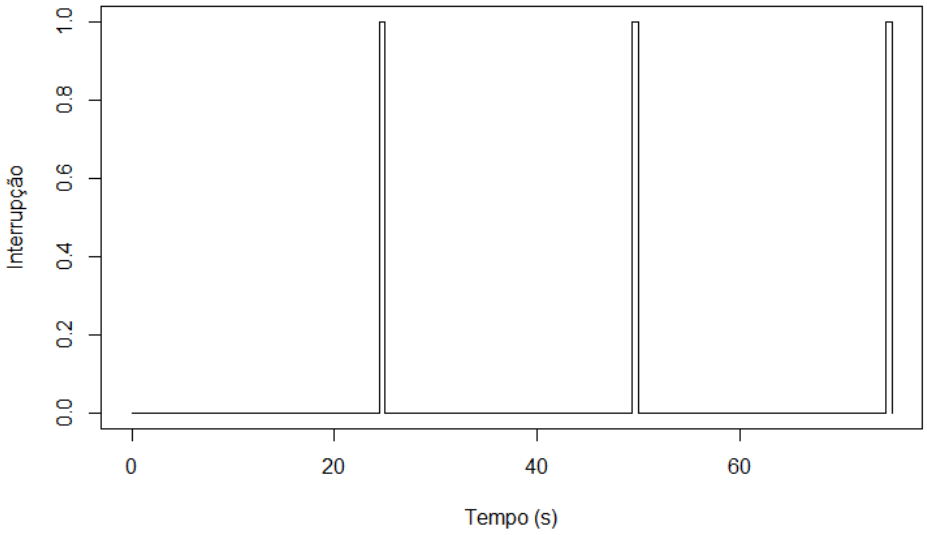


Figura 5.1: Interrupção do tráfego ao longo do tempo.

5.2 Métodos testados

Para comparar a qualidade do método proposto em relação aos métodos de detecção de anomalias mais comuns apliquei os diversos métodos às mesmas séries. Os métodos utilizados foram o kNN, o LOF e o método proposto. As redes neuronais não foram incluídas pelos motivos explicitados em 3.3. Todos estes métodos têm parâmetros que devem ser escolhidos de acordo com o tipo de problema em que serão utilizados.

5.2.1 Parâmetros kNN

O parâmetro essencial do kNN é o número de vizinhos mais próximos a comparar, k. Outros parâmetros menos comuns mas também eles com relevância são o tipo distância a calcular e se a escolha da medida é feita pelo vizinho k (mais distante dos k mais próximos) ou a média dos k vizinhos e no caso deste último, se o peso de cada um é igual ou difere conforme a distância. O tipo de distância usada é a euclidiana visto que se trata de comparação de valores unidimensionais e utilizou-se a média dos k vizinhos com pesos idênticos já que a solução alternativa se aproxima mais do LOF, método também ele a ser testado. Para o k, o valor deverá ser baixo para não ofuscar pequenas anomalias, mas não demasiado baixo uma vez que o ruído da série poderá levar a erros na classificação. A figura 5.3 mostra uma curva ROC onde se vê o resultado do kNN numa amostra do problema utilizando três k diferentes. O 4 foi aquele que obteve a melhor classificação e portanto a escolha para esse parâmetro.

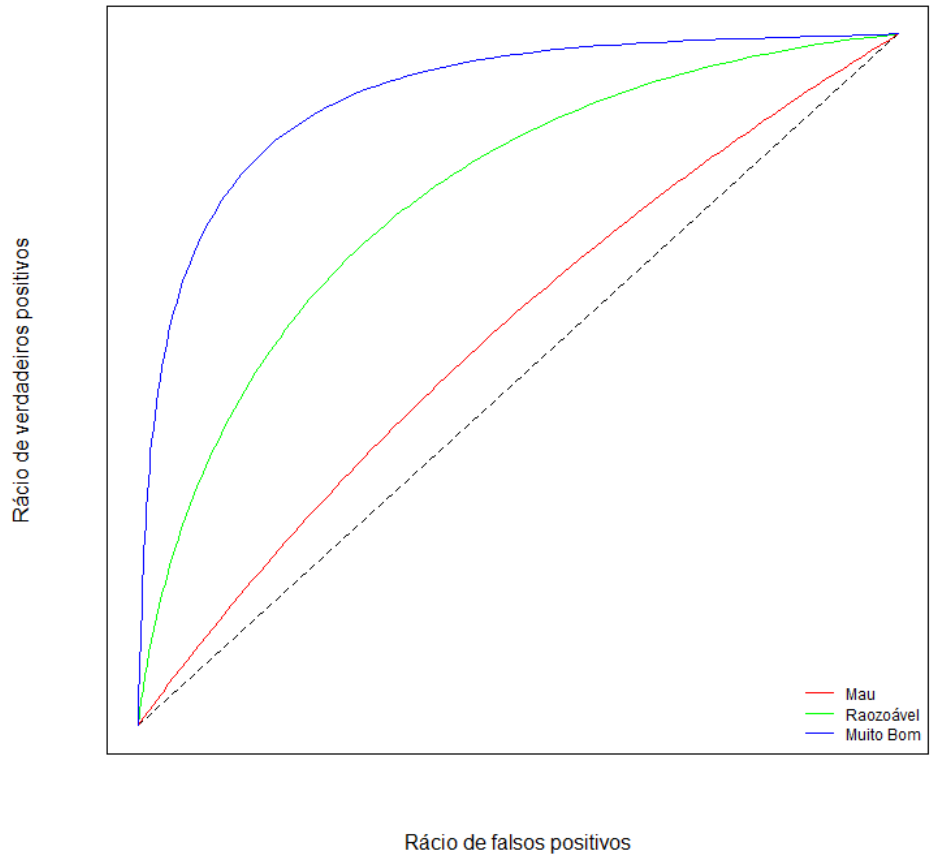


Figura 5.2: Exemplo de análise de curvas ROC.

5.2.2 Parâmetros LOF

O LOF tem como único parâmetro o número de vizinhos a procurar k. Sendo este algoritmo algo semelhante ao kNN, o número de vizinhos escolhido foi igualmente 4 pelas razões referidas em 5.2.1

5.2.3 Parâmetros DTW

O DTW contém 2 parâmetros principais. São estes o tipo de passo a realizar para a pesquisa de caminho e o *lower bounding* que é explorado mais adiante na secção 5.3. Para o tipo de passo escolheu-se o assimétrico, já que os passos simétricos não funcionam bem em pesquisas globais (neste caso de um segmento com uma série completa) e os passos mais complexos como os passos de *Rabiner*, não conseguem tão bons resultados em séries com alta variabilidade como as dos clientes [Gio09].

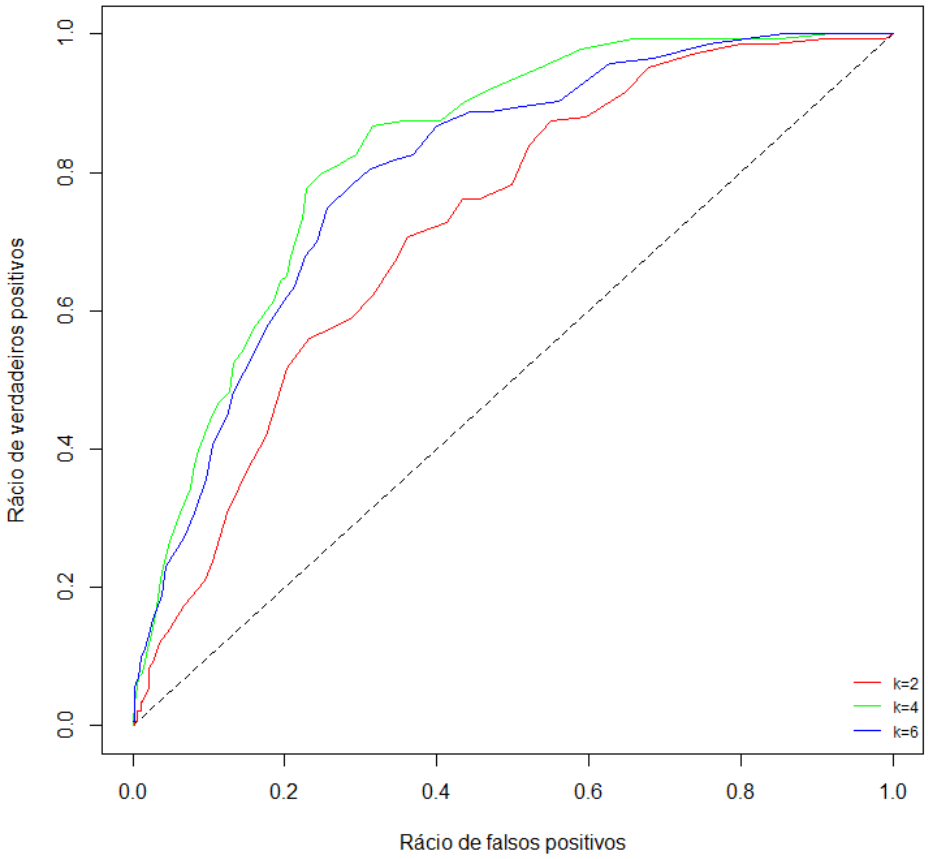
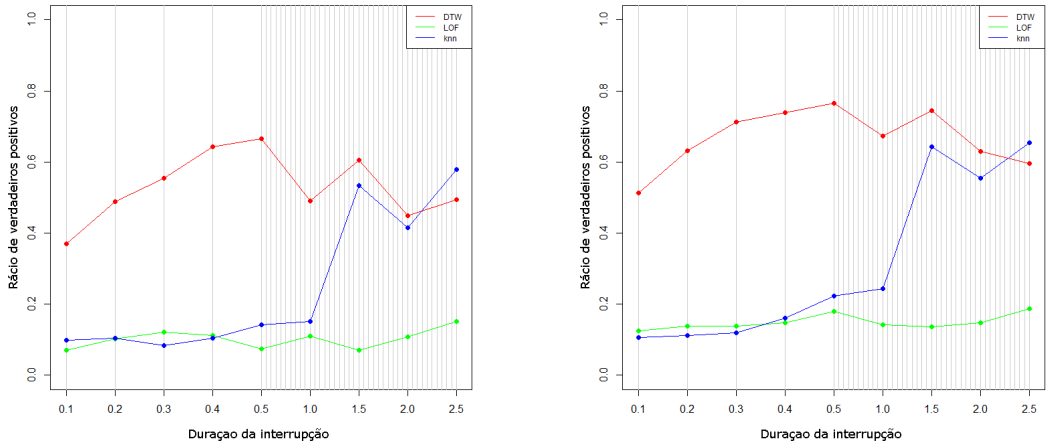


Figura 5.3: Comparação entre os valores de k a utilizar.

5.3 Resultados

Uma vez que a principal contribuição desta tese é um método de detecção de anomalias em servidores de *video on demand*, é importante que os resultados mostrem quão vantajoso é o método comparativamente aos métodos mais comuns e conhecidos. A avaliação dos métodos baseia-se na relação entre as anomalias classificadas e as não classificadas. Como foi dito anteriormente, os métodos como o kNN conseguem detectar anomalias com boa precisão somente quando estas são suficientemente expressivas. É interessante então que a comparação mostre a partir de que intensidade de anomalia é que o método proposto se torna vantajoso. As figuras que se encontram no anexo A mostram os resultados dos três métodos com anomalias que duram entre 0.1 segundos e 2 segundos. Para cada teste foram ativadas 290 interrupções. As figuras 5.4 mostram comparativamente para os três métodos, os rácios de verdadeiros positivos para 10% e 15% de falsos positivos. Estes rácios são obtidos encontrando um *threshold* na classificação que classifique erradamente 10% e 15% das observações classificadas como anomalias (falsos positivos). Foi dado mais ênfase às anomalias menos expressivas, entre 0.1 e 0.5 segundos, onde foi testado com

intervalos de 0.1 segundos e de 0.5 a 2.5 foram utilizados intervalos de 0.5 segundos.



(a) 10% de falsos positivos.

(b) 15% de falsos positivos.

Figura 5.4: Rácio de verdadeiros positivos para percentagem fixa de falsos positivos.

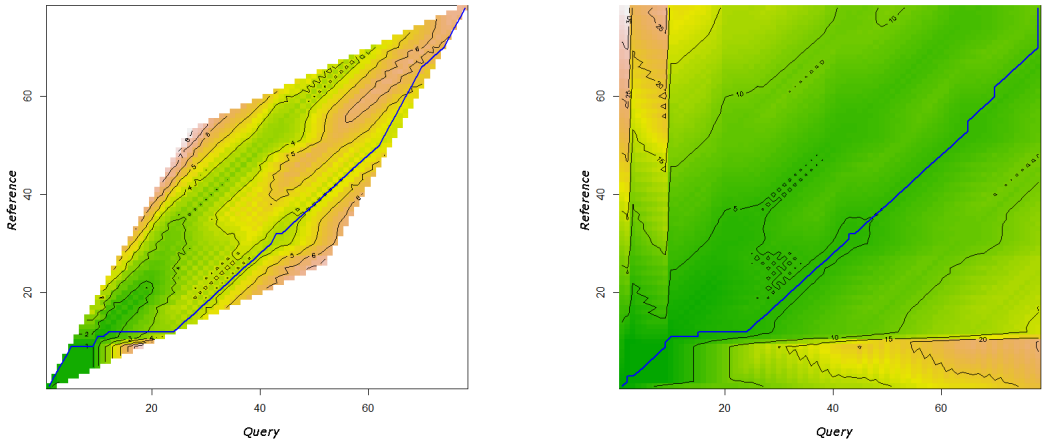
Os resultados mostram que com anomalias de 0.2 segundos a 1.0 segundo, o método proposto tem uma precisão aceitável e notavelmente melhor que as dos restantes algoritmos. Para anomalias de 0.1 a 0.5 segundos o algoritmo consegue encontrar $29.5\% \pm 4.3\%$, com 95% de confiança. No entanto, seria injusto concluir imediatamente que o método é claramente superior. O método proposto usa extensivamente o algoritmo de DTW que embora muito eficaz, é sabido que é um algoritmo pesado computacionalmente. É de esperar portanto que a performance do método seja pior que a dos métodos mais simples e porventura mais optimizados, uma vez que são mais estudados. É de notar também que o R, a plataforma utilizada para fazer os testes, por defeito não funciona em paralelo. A paralelização do método traria aumentos de *performance* significativos. A tabela 5.1 mostra o tempo que demorou cada um dos métodos a executar. Para todos os testes de performance foram feitas 8 execuções e utilizado um grau de 95% de confiança. O processador do computador utilizado foi um *Intel® Core™ i7-4700HQ @ 2.40GHz*.

Tabela 5.1: Tempos de execução dos métodos testados

Método	**Tempo de execução com 95% de confiança**
kNN	$1.2 \pm 0.1\ s$
LOF	$2.8 \pm 0.2\ s$
Método proposto	$435.4 \pm 35.1\ s$

Nota-se que para as mesmas séries, o método proposto demora demasiado tempo a executar. Isto condiciona completamente a sua utilização em tempo real remetendo-o para um método de análise posterior. Seria necessário então recorrer-se a optimizações. Uma das razões para isto acontecer é que o R, a linguagem usada para os testes, corre *single-threaded*. Seria possível paralelizar o método para obter melhor performance mas isso encontra-se fora do escopo da dissertação.

Em vez disso recorreu-se aos parâmetros do DTW. Sendo o DTW o principal *bottleneck* do algoritmo, este torna-se o principal alvo de optimizações. A optimização, de performance, mais comum do DTW é a de *lower bounding*. O *Lower Bounding* trata-se de colocar restrições no algoritmo de DTW de modo a que este não tenha de realizar todas as computações a partir de uma certa região. Isto pode levar a perda de precisão no método mas esta perda pode ser minimizada se a restrição utilizada for bem definida.



(a) Com lower bounding Itakura.

(b) Sem lower bounding.

Figura 5.5: DTW entre duas séries de clientes, com e sem lower bounding

Na figura 5.5a vê-se a região que é calculada, que neste caso tem a forma de um paralelogramo. Na figura 5.5b vê-se o calculo efetuado sem recurso a lower bounding. O resultado final (linha a azul) como se pode ver é semelhante visto que os caminhos mais curtos tendem a estar próximos da diagonal central. Outros métodos como o *SparseDTW* [ANCT09] ou *FastDTW* [SC07], são alternativas a considerar mas não serão analisados.

Tabela 5.2: Performance com e sem Lower Bounding

Método	**Tempo de execução**
Método sem lower bounding	$435.4 \pm 35.1\ s$
Método com Sakoe Chiba Window	$406.4 \pm 25.2\ s$

A variação de *performance* não foi muito significativa, cerca de 7.1%. uma vez que existem cálculos a ser eliminados é preciso analisar também a variação da precisão do método.

A imagem 5.6 mostra que para o mesmo conjunto de dados, o algoritmo teve um comportamento pior com o *lower bounding*. Esta variação na precisão torna o *lower bounding*, neste específico problema, pouco viável.

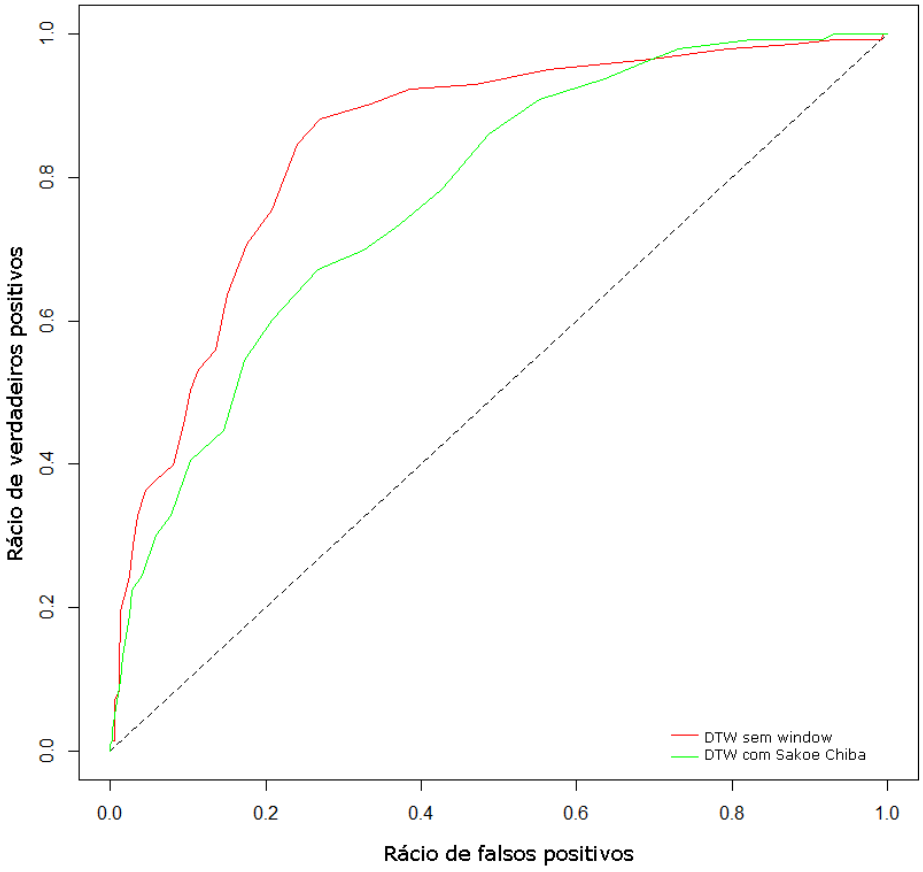


Figura 5.6: Resultados com e sem lower bounding.

Capítulo 6

Conclusões

O *Data Mining* é um tema muito estudado mas ainda com muito potencial de investigação e o *Data Mining* de séries temporais é de extrema relevância atualmente. Quanto à detecção de outliers, os estudos não se focaram muito nas séries temporais tornando este um tema ainda explorável. Para além disso o DTW, um algoritmo mais comummente associado ao processamento de sinal, mostrou ser uma ferramenta muito útil e versátil para a comparação entre séries temporais. O método proposto parece uma proposta viável para a detecção de anomalias pouco expressivas ainda que possa eventualmente necessitar de alguns ajustes em termos de performance. No entanto, as técnicas mais comuns provaram que funcionam em casos mais evidentes e a sua utilização pode ser colocada em causa conforme o problema em vista. Um tema sugerido nesta tese e que tem potencial, é a de indexação de séries na base de dados para consulta rápida. Este método seria uma opção viável para melhorar a performance do método. Uma vez que a pesquisa de séries seria mais rápida, a aplicação do método a outras situações que não *Video-on-Demand* poderia ser posta em consideração, uma vez que a comparação de séries poderia ser alargada a diversos tipos de comportamentos.

6.1 Outros métodos e melhoramentos futuros

O principal problema encontrado neste trabalho foi a performance do método. A *performance* demonstrada em relação aos métodos mais simples condiciona a sua implementação numa situação em tempo real a não ser que sejam feitas grandes otimizações. Uma das otimizações seria implementar, como referido anteriormente, um DTW mais eficiente como *FastDTW* ou *SparseDTW*, mas o próprio método em si poderia ser otimizado através da paralelização em CPU, já que o R por defeito corre em *Single-Thread* e as bibliotecas de paralelização são pobres. Uma vez que se tratam de pesquisas que podem ser feitas individualmente em que o resultado final é o valor mínimo encontrado, uma solução de *MapReduce* seria algo a explorar [DG08]. Um tema muito atual do *Data Mining* que se aplicava neste contexto é o de indexação que permite poupar

pesquisas à base de dados [KPC04, CPSK10]. Com estes métodos seria possível também criar um método com bases de dados maiores e mais diversas, logo mais genérico e aplicável a outras situações para além de *Video-on-Demand*.

Referências

[AJClZ11] Vijay Kumar Adhikari, Sourabh Jain, Yingying Chen e Zhi li Zhang. Reverse engineering the youtube video delivery cloud. In *In HotMD'11*, 2011.

[ANCT09] Ghazi Al-Naymat, Sanjay Chawla e Javid Taheri. Sparsedtw: A novel approach to speed up dynamic time warping. In *Proceedings of the Eighth Australasian Data Mining Conference - Volume 101*, AusDM '09, pages 117–127, Darlinghurst, Australia, Australia, 2009. Australian Computer Society, Inc. URL: http://dl.acm.org/citation.cfm?id=2449360.2449384.

[ASG04] M.J. Atallah, W. Szpankowski e R. Gwadera. Detection of significant sets of episodes in event sequences. In *Data Mining, 2004. ICDM '04. Fourth IEEE International Conference on*, pages 3–10, Nov 2004. doi:10.1109/ICDM.2004.10090.

[BBK14] M.H. Bhuyan, D.K. Bhattacharyya e J.K. Kalita. Network anomaly detection: Methods, systems and tools. *Communications Surveys Tutorials, IEEE*, 16(1):303–336, January 2014. doi:10.1109/SURV.2013.052213.00046.

[BHH01] Richard J. Bolton, David J. Hand e David J. H. Unsupervised profiling methods for fraud detection. In *Proc. Credit Scoring and Credit Control VII*, pages 5–7, 2001.

[BKNS00] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng e Jörg Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, May 2000. URL: http://doi.acm.org/10.1145/335191.335388, doi:10.1145/335191.335388.

[BR98] Simon Byers e Adrian E. Raftery. Nearest-neighbor clutter removal for estimating features in spatial point processes. *Journal of the American Statistical Association*, 93(442):577–584, 1998. URL: http://www.tandfonline.com/doi/abs/10.1080/01621459.1998.10473711, arXiv:http://www.tandfonline.com/doi/pdf/10.1080/01621459.1998.10473711, doi:10.1080/01621459.1998.10473711.

[CBK09] Varun Chandola, Arindam Banerjee e Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009. URL: http://doi.acm.org/10.1145/1541880.1541882, doi:10.1145/1541880.1541882.

[CDLDS13] O. Cigdem, T. De Laet e J. De Schutter. Classical and subsequence dynamic time warping for recognition of rigid body motion trajectories. In *Control Conference (ASCC), 2013 9th Asian*, pages 1–6, June 2013. doi:10.1109/ASCC.2013.6606405.

[CPSK10] A. Camerra, T. Palpanas, J. Shieh e E. Keogh. isax 2.0: Indexing and mining one billion time series. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 58–67, Dec 2010. doi:10.1109/ICDM.2010.124.

[DG08] Jeffrey Dean e Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008. URL: http://doi.acm.org/10.1145/1327452.1327492, doi:10.1145/1327452.1327492.

[Edg87] F.Y. Edgeworth. Xli. on discordant observations. *Philosophical Magazine Series 5*, 23(143):364–375, 1887. URL: http://dx.doi.org/10.1080/14786448708628471, arXiv:http://dx.doi.org/10.1080/14786448708628471, doi:10.1080/14786448708628471.

[FKL+08] Ada Wai-Chee Fu, Eamonn Keogh, Leo Yung Lau, Chotirat Ann Ratanamahatana e Raymond Chi-Wing Wong. Scaling and time warping in time series querying. *The VLDB Journal*, 17(4):899–921, July 2008. URL: http://dx.doi.org/10.1007/s00778-006-0040-z, doi:10.1007/s00778-006-0040-z.

[Gio09] Toni Giorgino. Computing and visualizing dynamic time warping alignments in r: The dtw package. *Journal of Statistical Software*, 31(7):1–24, 8 2009. URL: http://www.jstatsoft.org/v31/i07.

[HA02] Garry Hollier e Jim Austin. Novelty detection for strain-gauge degradation using maximally correlated components. pages 257–262, 2002.

[HTF01] Trevor Hastie, Robert Tibshirani e Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[JGP10] M. Jeihoonian, S.F. Ghaderi e M. Piltan. Modeling and comparing energy consumption in basic metal industries by neural networks and arima. In *Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference on*, pages 171–175, Oct 2010. doi:10.1109/CISIM.2010.5643670.

[JWH+13] M. Jarschel, F. Wamser, T. Hohn, T. Zinner e P. Tran-Gia. Sdn-based application-aware networking on the example of youtube video streaming. In *Software Defined Networks (EWSDN), 2013 Second European Workshop on*, pages 87–92, Oct 2013. doi:10.1109/EWSDN.2013.21.

[KCHP93] Eamonn Keogh, Selina Chu, David Hart e Michael Pazzani. Segmenting time series: A survey and novel approach. In *In an Edited Volume, Data mining in Time Series Databases. Published by World Scientific*, pages 1–22. Publishing Company, 1993.

[KM13] D.A. Kumar e S. Murugan. Performance analysis of indian stock market index using neural network time series model. In *Pattern Recognition, Informatics and Mobile Engineering (PRIME), 2013 International Conference on*, pages 72–78, Feb 2013. doi:10.1109/ICPRIME.2013.6496450.

[KPC04] Sang-Wook Kim, Sanghyun Park e Wesley W. Chu. Efficient processing of similarity search under time warping in sequence databases: An index-based approach. *Inf. Syst.*, 29(5):405–420, July 2004. URL: http://dx.doi.org/10.1016/S0306-4379(03)00037-1, doi:10.1016/S0306-4379(03)00037-1.

[LKLC03] Jessica Lin, Eamonn Keogh, Stefano Lonardi e Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th*

ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD '03, pages 2–11, New York, NY, USA, 2003. ACM. URL: http://doi.acm.org/10.1145/882082.882086, doi:10.1145/882082.882086.

[MGB12] A. Muscariello, G. Gravier e F. Bimbot. Unsupervised motif acquisition in speech via seeded discovery and template matching combination. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(7):2031–2044, Sept 2012. doi:10.1109/TASL.2012.2194283.

[MS03] Markos Markou e Sameer Singh. Novelty detection: a review—part 2:: neural network based approaches. *Signal Processing*, 83(12):2499 – 2521, 2003. URL: http://www.sciencedirect.com/science/article/pii/S0165168403002032, doi:http://dx.doi.org/10.1016/j.sigpro.2003.07.019.

[PG08] AS. Park e J.R. Glass. Unsupervised pattern discovery in speech. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(1):186–197, Jan 2008. doi:10.1109/TASL.2007.909282.

[Run12] T.A. Runkler. *Data Analytics: Models and Algorithms for Intelligent Data Analysis*. SpringerLink : Bücher. Springer-Verlag New York Incorporated, 2012. URL: http://books.google.pt/books?id=PaylsdmgkkIC.

[SC07] Stan Salvador e Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.*, 11(5):561–580, October 2007. URL: http://dl.acm.org/citation.cfm?id=1367985.1367993.

[She00] Colin Shearer. The crisp-dm model: The new blueprint for data mining. *Journal of Data Warehousing*, 5(4):13–22, 2000.

[SS06] Robert H. Shumway e David S. Stoffer. *Time Series Analysis and Its Applications: With R Examples (Springer Texts in Statistics)*. Springer, 2nd edition, May 2006. URL: http://www.worldcat.org/isbn/0387293175.

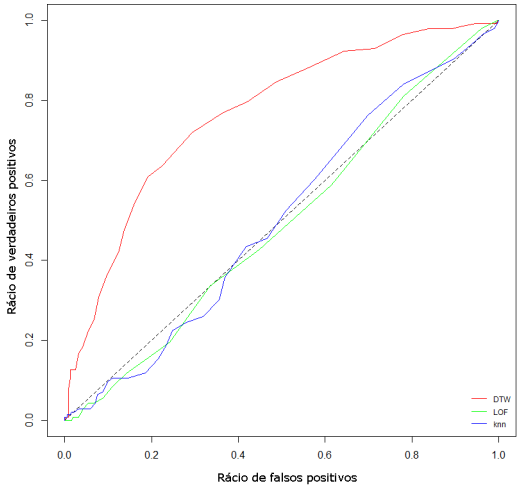
[WWG11] Yong Wang, Hao Wang e Zhi-Gang Gu. A survey of data mining softwares used for real projects. In *Open-Source Software for Scientific Computation (OSSC), 2011 International Workshop on*, pages 94–97, Beijing, China, Oct 2011. doi:10.1109/OSSC.2011.6184701.

REFERÊNCIAS

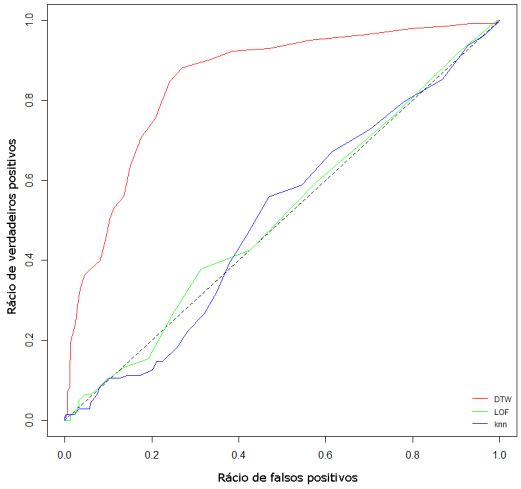
Anexo A

Resultados Comparativos

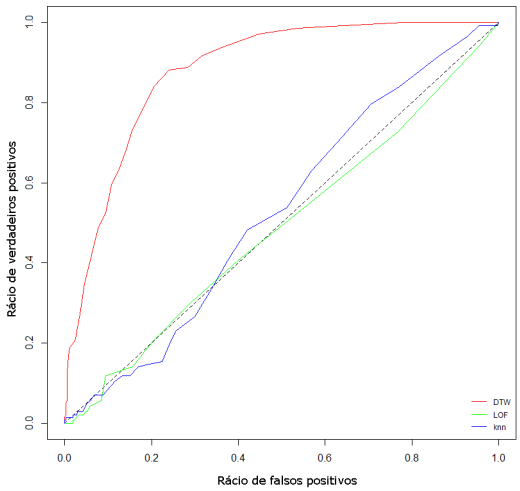
Aqui encontram-se as curvas ROC correspondentes às experiências efetuadas com os três métodos propostos.



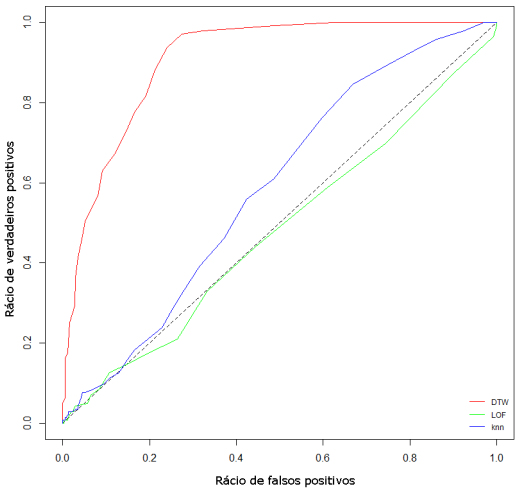
(a) Anomalia de 0.1 segundos.



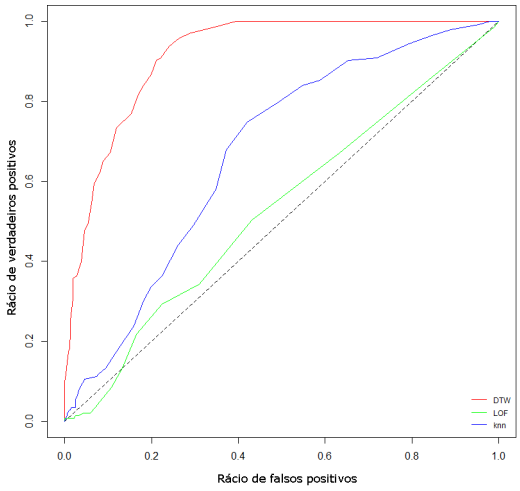
(b) Anomalia de 0.2 segundos.



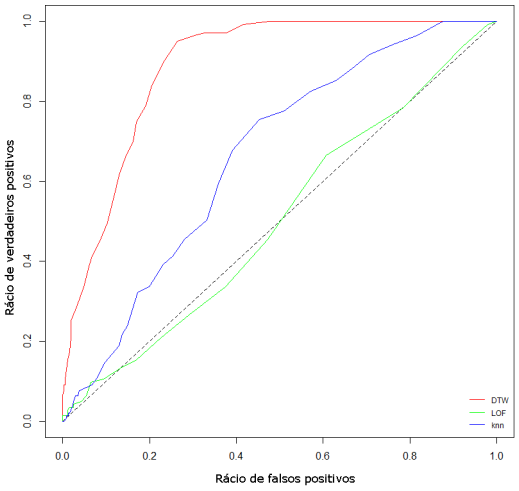
(a) Anomalia de 0.3 segundos.



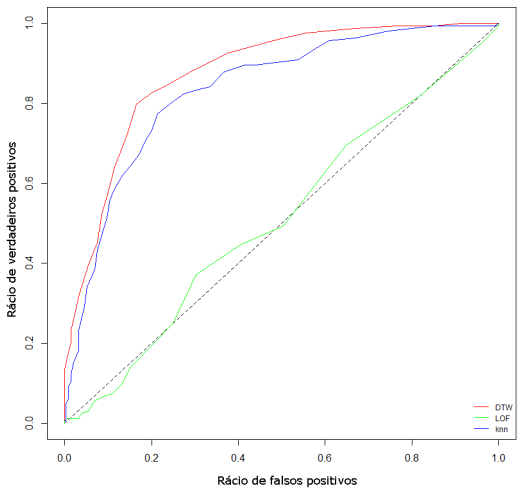
(b) Anomalia de 0.4 segundos.



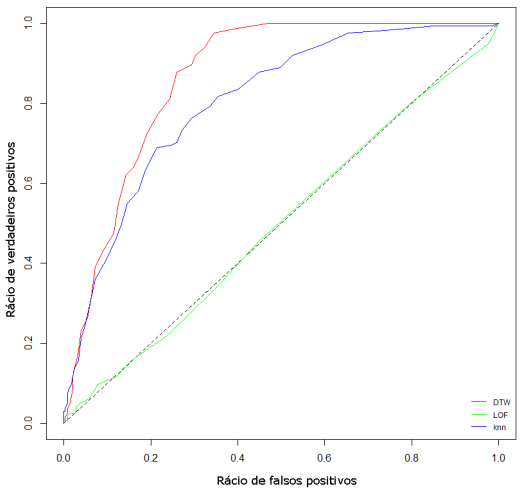
(a) Anomalia de 0.5 segundos.



(b) Anomalia de 1 segundo.



(a) Anomalia de 1.5 segundos.



(b) Anomalia de 2 segundos.