Amostragem e medidas de qualidade de shapelets

Lucas Schmidt Cavalcante

Lucas Schmidt Cavalcante

Amostragem e medidas de qualidade de shapelets

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

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Gustavo Enrique de Almeida Prado Alves Batista

USP – São Carlos
Fevereiro de 2016

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

C376a

Cavalcante, Lucas Schmidt
Amostragem e medidas de qualidade de shapelets / Lucas Schmidt Cavalcante; orientador Gustavo Enrique de Almeida Prado Alves Batista. - São Carlos - SP, 2016.
81 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2016.

1. Classificação de séries temporais. 2. Transformada *shapelet*. 3. Medidas de qualidade. 4. Amostragem aleatória. I. Batista, Gustavo Enrique de Almeida Prado Alves, orient. II. Título.

Lucas Schmidt Cavalcante

Shapelets sampling and quality measurements

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *EXAMINATION BOARD PRESENTATION COPY*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Gustavo Enrique de Almeida Prado Alves Batista

USP – São Carlos
February 2016

Aos meus pais,
Maurício e Elizabeth.

AGRADECIMENTOS

Agradecimentos aos professores João do Espirito Santo Batista Neto e Gustavo Enrique de Almeida Prado Alves Batista, e ao Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo.

“There is no universal solution,
but there is a universal process to find appropriate local solutions.”

(Carl E. Taylor)

RESUMO

CAVALCANTE, L. S.. **Amostragem e medidas de qualidade de shapelets**. 2016. 81 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Uma série temporal é uma sequência ordenada pelo tempo de valores reais. Dado que inúmeros fenômenos do dia-a-dia podem ser representados por séries temporais, há grande interesse na mineração de dados temporais, em especial na tarefa de classificação. Recentemente foi introduzida uma nova primitiva de séries temporais chamada *shapelet*, que é uma subsequência que permite a classificação de séries temporais de acordo com padrões locais. Na transformada *shapelet* estas subsequências se tornam atributos em uma matriz de distância que mede a dissimilaridade entre os atributos e as séries temporais. Para obter a transformada é preciso escolher alguns *shapelets* dos inúmeros possíveis, seja pelo efeito de evitar *overfitting* ou pelo fato de que é computacionalmente caro obter todos. Sendo assim, foram elaboradas medidas de qualidade para os *shapelets*. Tradicionalmente tem se utilizado a medida de ganho de informação, porém recentemente foi proposto o uso da *f-statistic*, e nós propomos neste trabalho uma nova denominada *in-class transitions*. Em nossos experimentos demonstramos que a *in-class transitions* costuma obter a melhor acurácia, especialmente quando poucos atributos são utilizados. Além disso, propomos o uso de amostragem aleatória nos *shapelets* para reduzir o espaço de busca e acelerar o processo de obtenção da transformada. Contrastamos a abordagem de amostragem aleatória contra uma em que só são exploradas *shapelets* de determinados tamanhos. Nossos experimentos mostraram que a amostragem aleatória é mais rápida e requer a computação de um menor número de *shapelets*. De fato, obtemos os melhores resultados ao amostrarmos 5% dos *shapelets*, mas mesmo a uma amostragem de 0,05% não foi possível notar uma degradação significante da acurácia.

Palavras-chave: Classificação de séries temporais, Transformada *shapelet*, Medidas de qualidade, Amostragem aleatória.

ABSTRACT

CAVALCANTE, L. S.. **Amostragem e medidas de qualidade de shapelets**. 2016. 81 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

A time series is a time ordered sequence of real values. Given that numerous daily phenomena that can be described by time series, there is a great interest on its data mining, specially on the task of classification. Recently it was introduced a new time series primitive called *shapelets*, that is a subsequence that allows the classification of time series by local patterns. On the *shapelet* transformation these subsequences turn into attributes in a distance matrix that measures the dissimilarity between these attributes and the time series. To obtain the *shapelet* transformation it is required to choose some *shapelets* among all of the possible ones, be it to avoid *overfitting* or because it is too computationally expensive to obtain everyone. Thus, some *shapelet* quality measurements were created. Traditionally the information gain has been used as the default measurement, however, recently it was proposed to use the *f-statistic* instead, and in this work we propose a new one called *in-class transitions*. On our experiments it is shown that usually the *in-class transitions* achieves the best accuracy, specially when few attributes are used. Moreover, we propose the use of random sampling of *shapelets* as a way to reduce the search space and to speed up the process of obtaining the *shapelet* transformation. We contrast this approach with one that explores only *shapelets* that have a specific length. Our experiments show that random sampling is faster and requires fewer *shapelets* to be computed. In fact, we got the best results when we sampled 5% of the *shapelets*, but even at a rate of 0.05% it was not possible to detect a significant degradation of the accuracy.

Key-words: Time series classification, *Shapelet* transformation, Quality measurements, Random sampling.

LISTA DE ILUSTRAÇÕES

LISTA DE ALGORITMOS

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

Uma série temporal é uma sequência ordenada de valores reais (LINES *et al.*, 2012), no qual a ordem costuma ser dada pelo tempo. Dado que inúmeros fenômenos do dia-a-dia podem ser representados por séries temporais, há grande interesse na mineração de dados temporais. Por exemplo, em medicina, foi realizado um trabalho para indução e avaliação de regras do tipo *se-então* em um banco de dados de pacientes com hepatite crônica, que tiveram seu sangue e urina analisados ao longo de vários anos (ABE; YAMAGUCHI, 2005); em genética, o tempo é um fator importante a ser levado em consideração na análise de expressão gênica (ANDROULAKIS *et al.*, 2005); na botânica, séries temporais foram empregadas com sucesso para classificação de plantas (YE; KEOGH, 2009), tanto pela observação de dados temporais quanto pela transformação da forma do contorno das folhas em séries temporais utilizando um truque de representação; em metereologia, séries temporais são usadas como atributos em árvores de decisão para predição de formação de tornados (MCGOVERN *et al.*, 2007); em um outro exemplo, o comportamento recente do vento é utilizado para se buscar por um comportamento passado similar que possa ser utilizado para estimar quanto de energia eólica poderá ser gerado no momento (KAMATH; FAN, 2012); em entretenimento, séries temporais são empregadas para criação de animações a partir de *live-action videos* (CELLY; ZORDAN, 2004); em astronomia, séries temporais extraídas a partir da intensidade de luz de estrelas são analisadas para detecção de comportamento anômalos (YE *et al.*, 2008); na indústria de manufatura séries temporais são extraídas de sensores colocados na linha de produção para averiguar a qualidade de produção (PATRI *et al.*, 2014); entre muitos outros domínios.

Os exemplos de uso citados anteriormente ilustram algumas das principais tarefas de mineração em séries temporais, que são: classificação, agrupamento, descoberta de anomalias e de padrões recorrentes. Este trabalho está-se interessado sobretudo na classificação de séries temporais.

1.1 Justificativa e Motivação

No problema supervisionado de classificação de séries temporais se deseja rotular novas séries temporais a partir de um conjunto de treinamento no qual as séries temporais possuem rótulo conhecido. Por muitos anos a comunidade cientifica usou o classificador de k-vizinhos mais próximos (kNN) para classificar séries temporais, o qual em conjunto com a medida de dissimilaridade *Dynamic Time Warping* (DTW) se mostrou difícil de ser superado (DING *et al.*, 2008). No entanto, Ye e Keogh (2009) apresentaram uma nova primitiva de séries temporais chamada *shapelet*, que em conjunto com o algoritmo de classificação de árvore de decisão apresentou acurácia superior ao classificador de vizinhos mais próximos em diversos conjuntos de dados.

Shapelet é uma subsequência de uma série temporal que é a mais representativa de uma classe. Esta representação advém da sua criação e avaliação. Inicialmente todas as subsequências possíveis são geradas, então para cada uma delas avalia-se a sua similaridade com cada uma das séries, no qual a similaridade entre uma subsequência e uma série temporal de maior número de amostras é dada pela menor distância euclidiana encontrada ao se deslizar a subsequência ao longo da série temporal. Uma subsequência de alta representatividade é aquela que possui alta similaridade em séries de uma determinada classe mas baixa em séries de outras classes. Este processo de extração e avaliação de subsequências é embutido em uma árvore de decisão, na qual a representatividade de uma subsequência é avaliada de forma quantitativa pelo ganho de informação. Isto descreve a grosso modo o classificador proposto por Ye e Keogh (2009), aqui denotado por árvore de decisão *shapelets*. O poder desse classificador vem de suas subsequências:

- Seja um conjunto de dados no qual o padrão que distingue as classes seja uma pequena subsequência em relação ao tamanho da série temporal, e que tais séries possuam ruído, então ao se medir a dissimilaridade usando a série toda o ruído pode dominar a dissimilaridade ofuscando qualquer diferença causada pelo padrão. Nesses casos o classificador de vizinhos mais próximos possui acurácia equivalente a de palpites aleatórios.

- Ao se deslizar a subsequência ao longo da série temporal para computar a dissimilaridade tem como efeito prático não exigir que a subsequência que distingue as classes apareça sempre na mesma posição nas séries temporais.

- Árvores de decisão possuem grande interpretabilidade e subsequências ainda mais. Ao final do processo de indução da árvore de decisão é possível inspecionar as subsequências utilizadas nos nós da árvore, tanto pelo padrão que representam como de quais e de onde nas séries temporais elas foram extraídas.

Lines *et al.* (2012) vai além com a ideia de usar subsequências em uma árvore de decisão e introduz a transformada *shapelet*. A transformada consiste em escolher as subsequências

de maior ganho de informação e transformá-las em atributos, e então preencher uma matriz de distância desses atributos às séries temporais. Esta transformada dissocia o processo de descoberta e extração de subsequências do processo de classificação, e assim permite o uso de inúmeros classificadores. Nos experimentos conduzidos pelos autores, no qual induziram os classificadores *Naive Bayes*, kNN, *Rotation Forest*, *Random Forest*, *Bayesian Networks* e *Support Vector Machines* (SVM) sob a transformada, eles notaram que os classificadores mais complexos (*Rotation Forest*, *Random Forest*, *Bayesian Networks* e em especial o SVM) possuíam as maiores acurácias, que por sua vez também obtiveram maior acurácia em relação a árvore de decisão *shapelets*. Os autores também fizeram dois outros curtos experimentos: no primeiro eles compararam a acurácia da árvore de decisão *shapelets* contra uma árvore de decisão induzida sob a transformada e concluíram que não havia diferença entre os resultados dos classificadores; o segundo, também realizado em uma árvore de decisão sob a transformada, consistiu em trocar o uso do ganho de informação pelo valor da *f-statistic* para selecionar as subsequências, pela hipótese de que o ganho de informação pode ser inadequado para a tarefa de descrever a distribuição das séries de inúmeras classes, no qual notaram que apesar da *f-statistic* ser mais rápida, o ganho de informação apresentou uma acurácia ligeiramente superior, logo descartaram a troca.

Lines e Bagnall (2012) também questionaram o uso do ganho de informação como a medida de qualidade da subsequência. Em seu trabalho o ganho de informação é comparado a duas outras medidas de qualidade na árvore de decisão *shapelets*: a mediana de Mood e o teste não-paramétrico de Kruskal-Wallis. Os resultados mostraram que em termos de acurácia não há diferença significativa entre essas medidas, mas estas duas costumam acelerar o processo de avaliação das subsequências em quase 20% em relação ao ganho de informação.

Por fim, Lines e Bagnall (2012) e de Lines *et al.* (2012) colaboraram em um novo trabalho no qual sumarizaram suas descobertas e realizaram novos experimentos. Dentre os experimentos, foi reavaliado o uso de medidas de qualidade alternativas na árvore de decisão *shapelets*, no qual a *f-statistic* obteve a melhor acurácia (sem significância estatística em relação as outras) e o menor tempo de execução, concluindo, na opinião deles, de que a *f-statistic* deve ser a escolha padrão para a medida de qualidade. Em outro experimento confirmaram que em termos de acurácia é melhor utilizar a transformada em conjunto com classificadores mais complexos do que a árvore de decisão *shapelets*. No entanto, ainda há inúmeras questões em aberto em relação a transformada. Por exemplo, a definição da quantidade de atributos a ser usado ainda não foi respondida. Também não há experimentos extensivos que avaliam as medidas de qualidade de subsequências no domínio da transformada. E o principal problema de técnicas que envolvem subsequências permanece sem solução satisfatória, pois a quantidade de subsequências possíveis de um conjunto de dados é enorme, e a avaliação de cada uma delas é um processo custoso, logo em seus trabalhos eles realizaram experimentos em subsequências de tamanho restrito (HILLS *et al.*, 2014).

1.2 Lacunas & Objetivos

Dada a nossa revisão bibliográfica sobre a classificação de séries temporais nós identificamos 3 lacunas no processo de classificação pela transformada *shapelet*. A primeira das lacunas está no fato de que os experimentos apresentados até o momento foram em um conjunto reduzido do total de *shapelets* por questões de tempo de execução. Por exemplo, Hills *et al.* (2014) propôs um algoritmo que restringe a busca por bons *shapelets* àqueles que possuam uma quantidade de amostras entre um valor mínimo e máximo. A nossa crítica é que sem experimentos que utilizem todos os *shapelets* qualquer avaliação, seja de acurácia ou de tempo de execução, de métodos que reduzem o espaço de busca dos *shapelets* tem baixo valor científico. Sendo assim, nosso primeiro objetivo é realizar experimentos que utilizem todos os *shapelets*:

> *Hills et al. (2014) conduziram experimentos com restrição ao tamanho das subsequências extraídas, pois computar todas é um processo caro. No entanto, sem ter o resultado da computação total a avaliação de métodos que fazem um speedup do processo de classificação por ignorar certas subsequências se torna nulo. Um ground-truth é necessário.*

A segunda lacuna é relacionada a primeira. Como dito, Hills *et al.* (2014) emprega um algoritmo para estimar automaticamente o valor de quantidade máxima e mínima de amostras nos *shapelets* a serem explorados, porém o método mais simples de amostragem aleatória parece ter sido esquecido. Logo, nosso segundo objetivo é realizar experimentos que fazem a amostragem aleatória dos *shapelets* em diferentes níveis de proporção e avaliar como isso afeta a acurácia e o tempo de execução do processo de classificação:

> *Avaliar como a amostragem aleatória de shapelets em diferentes níveis de proporção afeta a acurácia e o tempo de execução do processo de classificação pela transformada shapelet. Em especial, fazer um comparativo da amostragem aleatória com o método de Hills et al. (2014) de estimação de valor de mínimo e máximo para a quantidade de amostras nos shapelets.*

Por fim, um passo de crucial importância na transformada é a escolha de quais *shapelets* irão compo-la. Essa escolha é sensível tanto a quantidade quanto a qual medida de qualidade é usada para selecionar os *shapelets*. Neste trabalho nós abordamos somente a segunda questão, sobre o uso de medidas de qualidade. Historicamente se tem utilizado o ganho de informação, mas tanto os experimentos de Lines e Bagnall (2012) quanto de Hills *et al.* (2014) em uma árvore de decisão *shapelets* mostraram que existem outras medidas que geram modelos sem degradação da acurácia mas com maior velocidade. Porém, Hills *et al.* (2014) foram além e somente com base nos resultados da árvore de decisão *shapelets* decidiram recomendar o uso da *f-statistic* em favor do ganho de informação na transformada. Nós acreditamos que os seus experimentos

não são conclusivos o suficiente para tal decisão e por isso temos por objetivo final realizar os primeiros experimentos extensivos de avaliação de medidas de qualidade na transformada. Em especial, iremos avaliar as medidas de qualidade já propostas, ganho de informação e *f-statistic*, com a nossa proposta, denominada *in-class transitions*, que é baseada na de ganho de informação e explora diretamente a ordenação das séries de diferentes classes em relação ao *shapelet*:

> *Realizar os primeiros experimentos que visam avaliar como as medidas de qualidade de shapelets influenciam a acurácia no domínio da transformada shapelet. Além disso, avaliar a nossa medida de qualidade proposta, denominada in-class transitions, que tem por base a de ganho de informação e explora diretamente a ordenação das séries de diferentes classes em relação ao shapelet.*

1.3 Principais Contribuições

Em nosso trabalho fizemos as primeiras avaliações extensivas sobre como as medidas de qualidade e a amostragem aleatória de *shapelets* afetam a acurácia dos classificadores. No experimento em que comparamos as medidas de qualidade quando todos os *shapelets* são utilizados e ao variarmos a quantidade de atributos na transformada, notamos que a *f-statistic* possui a pior acurácia, e tanto a *in-class transitions* quanto a de ganho de informação possuem acurácias semelhantes, porém, a *in-class transitions* domina quando poucos atributos são utilizados. É um resultado surpreendente e vai de encontro a recomendação de Hills *et al.* (2014). Ademais, em nossos experimentos em que o espaço de busca dos *shapelets* é reduzido pelo uso de amostragem aleatória em diferentes níveis de proporção, nós notamos que os melhores resultados ocorrem em cerca de 5% de amostragem, sendo até mesmo melhor ao caso em que não é feita nenhuma redução no espaço de busca. Nós acreditamos que isso ocorreu pois a amostragem aleatória forçou uma diversificação nos atributos resultando em uma transformada com poucos atributos redundantes, auxiliando assim o classificador. Além disso, mesmo ao se amostrar a um nível de 0.5% não é possível notar uma grande degradação na acurácia dos modelos finais, sendo assim a amostragem aleatória se mostra como uma ótima maneira para reduzir o tempo de execução. Ainda sobre amostragem aleatória, ao fazemos uma comparação direta com a técnica de redução de espaço de busca por restrição na quantidade de amostras nos *shapelets* recomendado por Hills *et al.* (2014), notamos que embora a amostragem aleatória tenha somente uma pequena influência positiva na acurácia, a técnica de Hills *et al.* (2014) costuma oferecer intervalos largos de cerca de 30% do total de *shapelets* contra nossa recomendação de 5%. Por fim, ao compararmos o nosso modelo de classificação (uso da *in-class transitions* com amostragem aleatória de 5%) contra o modelo de Hills *et al.* (2014), o nosso modelo possui maior acurácia na maioria das vezes.

1.4 Organização do Trabalho

Este trabalho está organizado do seguinte modo: Capítulo 2 é dedicado a introduzir as notações que serão usadas ao longo de todo o trabalho e também faz a formalização do que são as séries temporais, incluindo a sua representação e suas medidas de dissimilaridades; Capítulo 3 tem como objetivo detalhar os algoritmos de classificação utilizados neste trabalho, que são o algoritmo de *Nearest Neighbors*, o de árvore de decisão, o *Random Forest* e o *Support Vector Machines*; Capítulo 4 faz uma revisão dos principais trabalhos relacionados a *shapelets*, incluindo a árvore de decisão *shapelet*, a transformada *shapelet* e as nossas propostas; Capítulo 5 contêm todos os experimentos e resultados deste trabalho, bem como uma análise sobre eles; por fim, o Capítulo 6 finaliza este trabalho por incluir as nossas conclusões e os possíveis trabalhos futuros.

CAPÍTULO

2

SÉRIES TEMPORAIS

2.1 Considerações Iniciais

Este capítulo tem por objetivo definir séries temporais e estabelecer as notações que serão usadas em todo o trabalho. Tanto a definição formal de uma série temporal quanto as notações sobre séries e operações nelas aplicadas são dadas na Seção 2.2.

Como dito previamente, séries temporais são uma sequência de valores reais, e pela exceção de um algoritmo, neste trabalho todos os algoritmos fazem operações sobre uma sequência numérica. A exceção é um algoritmo aproximado para a árvore de decisão *shapelets* que utiliza uma transformação da sequência numérica em uma simbólica para acelerar a computação. Estas duas representações, a numérica e simbólica, incorrem em diferentes estruturas de dados e medidas de dissimilaridade. A estrutura de dados e a medida de dissimilaridade para o caso numérico é abordada na Seção 2.3, enquanto que a Seção 2.4 aborda o caso simbólico, incluindo o processo de transformação.

2.2 Definições e Notações

Uma série temporal T é uma sequência ordenada de valores reais (LINES *et al.*, 2012), no qual tal ordem deve ser respeitada pois contém informação vital sobre o fenômeno a ser estudado. Uma série temporal T de m amostras é representada por $T = \{t_1, t_2, \ldots, t_m\}$, no qual cada amostra t_i pode ser um valor numérico ou, no caso em que tenha passado por um processo de simbolização, um símbolo, como é detalhado na Seção 2.3 sobre representação numérica e na Seção 2.4 sobre representação simbólica.

A partir de uma série temporal T é possível extrair uma subsequência temporal S, que é uma sequência contínua de amostras de T (RAKTHANMANON; KEOGH, 2013). Por exemplo, uma subsequência de 4 amostras com início na 5ª amostra de T é representada por

$S = \{t_5, t_6, t_7, t_8\}$. Por mais que S seja uma subsequência, ela também é uma série temporal.

Além disso, como este trabalho está centrado na classificação de séries temporais, cada série temporal T possui um rótulo de classificação c, formando assim uma tupla $\langle T, c \rangle$. Apesar do rótulo ser definido pela aplicação, neste trabalho, por simplicidade, um rótulo é um valor inteiro entre 1 e C. Ademais, definimos um Conjunto de Dados, CD, como um conjunto de n tuplas: $CD = \{\langle T_1, c_1 \rangle, \langle T_2, c_2 \rangle, \langle T_3, c_3 \rangle, \ldots, \langle T_n, c_n \rangle\}$.

A comparação entre duas séries temporais quaisquer, denotadas aqui por S e T, é feita por uma medida de distância denotada por $dist(S,T)$. O conjunto de distâncias de uma série temporal S à todas as séries de um conjunto de dados é denotado por $D_S = \{d_{S,1}, d_{S,2}, d_{S,3}, \ldots, d_{S,n}\}$, no qual $d_{S,i}$ é necessariamente um valor numérico e por isso podemos definir $\bar{D_S}$ como sendo a média desses valores.

Por fim, denotaremos por $CD_{|c_i}$ e $D_{S|c_i}$ as partições desses conjuntos de acordo com uma classe c_i qualquer, e por $|CD_{|c_i}|$ e $|D_{S|c_i}|$ as respectivas cardinalidades dessas partições.

2.3 Representação Numérica

Na representação numérica cada amostra t_i de uma série temporal T é um valor real, $t_i \in \mathbb{R}$. Para esta representação existem inúmeras medidas de dissimilaridade, sendo que uma das mais utilizada é a distância euclidiana. De fato, no estudo de Giusti e Batista (2013) foram analisadas 48 medidas de dissimilaridade em 42 conjuntos de dados, e destas somente três conseguiram ter uma acurácia superior (com significância estatística) em relação a distância euclidiana. Na Seção 2.3.1 será detalhada a distância euclidiana e a *Dynamic Time Warping* (DTW), que é uma das que possuiu acurácia superior. Além disso, também iremos justificar o uso da distância euclidiana neste trabalho.

2.3.1 Medida de Dissimilaridade

A distância euclidiana é uma métrica muito utilizada na área de séries temporais, e apesar de ser sensível a distorções no eixo temporal (KEOGH, 2002; KEOGH; KASETTY, 2003; RATANAMAHATANA; KEOGH, 2004a), ela possui vantagens como respeitar a desigualdade triangular, ser de fácil implementação e ter complexidade temporal linear em relação ao tamanho da série temporal. A distância euclidiana entre duas séries temporais quaisquer T e S de tamanho m é a raiz-quadrada da soma do quadrado da diferença, como mostrado pela Equação 2.1.

$$dist(S,T) = \sqrt{\sum_{i=1}^{m} (s_i - t_i)^2} \tag{2.1}$$

Mesmo com sua sensibilidade à distorções no eixo temporal, foi demonstrado que na prática a distância euclidiana tem boa performance em relação a outras medidas propostas

(KEOGH; KASETTY, 2003; GIUSTI; BATISTA, 2013). No entanto, é recomendado que ou se aplique um pré-processamento na série temporal para remover tais distorções ou que outra medida mais robusta seja utilizada, por exemplo, a DTW (BATISTA *et al.*, 2014).

Um dos pré-processamentos disponíveis, e que por nós é utilizado antes de medir a distância entre duas séries temporais, é a *z-normalização* das amostras de S e T. A Figura 1 ilustra a necessidade desse pré-processamento, no qual em uma série temporal existem duas subsequências similares, porém, a subsequência à direita possui um *offset* positivo e uma menor amplitude em relação a subsequência à esquerda. É crucial a remoção dessas distorções, pois pequenas diferenças em amplitude ou *offset* logo dominam a dissimilaridade entre séries (MUEEN; KEOGH; YOUNG, 2011). Para *z-normalizar* uma série T é preciso, para cada amostra t_i, subtrair a média e dividir pelo desvio padrão da série temporal, como mostra a Equação 2.2.

$$t_i = \frac{t_i - \bar{T}}{\sigma_T}, \forall t_i \in T \tag{2.2}$$

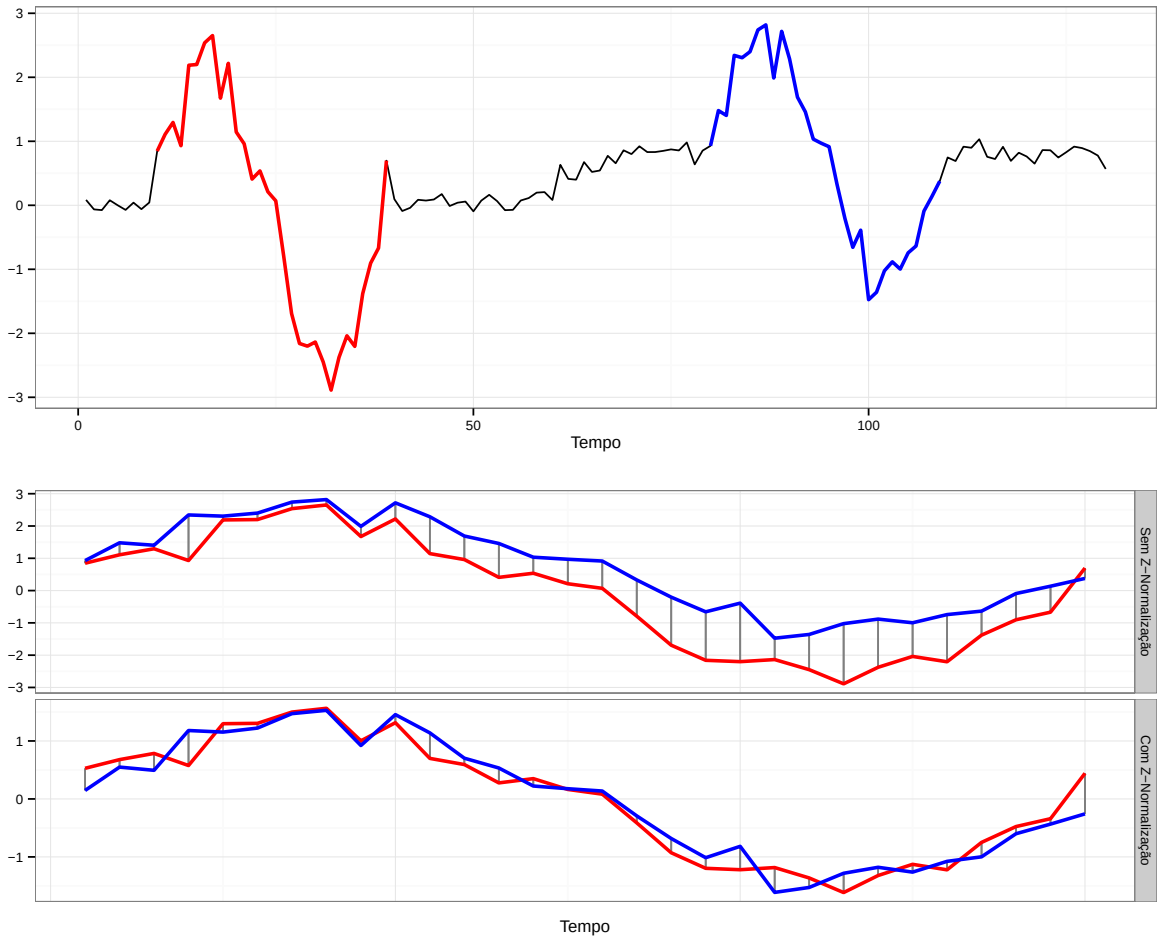

Figura 1 – Acima uma série temporal que possui duas subsequências similares, no qual a subsequência à direita possui um *offset* positivo e uma amplitude menor em relação a subsequência à esquerda. Abaixo mostra-se como o uso da *z-normalização* remove tais distorções que podem acentuar a dissimilaridade entre as séries pela distância euclidiana (as linhas verticais ilustram a distância euclidiana, que é a raiz quadrada da soma do quadrado da magnitude dessas linhas).

Além da distorção causada por diferenças entre *offset* e amplitude que deve ser removida, a distância euclidiana apresenta duas rigidezas que em alguns casos é indesejável. A primeira rigidez é que a distância euclidiana da Equação 2.1 só está definida para séries temporais de mesmo tamanho. Entretanto, em algumas situações existe a necessidade de se comparar séries temporais de diferentes tamanhos. Por exemplo, em *shapelets* (YE; KEOGH, 2009) foi definida a distância entre séries de diferentes tamanhos da seguinte maneira: seja S uma série temporal de tamanho m', e T outra série de tamanho m, sendo que $m' < m$, então a distância euclidiana destas séries é dada pela menor distância obtida ao se deslizar a série de menor tamanho pela série de maior tamanho. Além disso, para tornar viável a comparação de distâncias entre quaisquer pares de séries, é recomendado a normalização pelo tamanho da menor série temporal (MUEEN; KEOGH; YOUNG, 2011). Todo este processo é formalizado pela Equação 2.3.

$$dist(S,T) = min_{0 \leq i < m-m'} \left\{ \sqrt{\frac{1}{m'} \sum_{j=1}^{m'} (s_j - t_{j+i})^2} \right\} \tag{2.3}$$

A segunda rigidez está relacionada ao fato de que a distância euclidiana da Equação 2.1 só compara amostras tomadas no mesmo tempo. Porém, algumas aplicações requerem uma maior flexibilidade ao permitir uma comparação de s_i com t_j ($i \neq j$), pois uma das séries pode ter tido algum segmento "encurtado" ou "esticado" no eixo temporal, logo é outra distorção que deve ser levada em consideração. A medida de dissimilaridade *Dynamic Time Warping* contorna este problema por determinar um mapeamento de comparações entre as amostras de modo a minimizar essas distorções. Este mapeamento deve respeitar as seguintes condições (RATANAMAHATANA; KEOGH, 2004b):

Restrição de fronteira: o mapeamento deve compreender todas as amostras das duas séries temporais, ou seja, começa nas primeiras amostras (1, 1) e termina nas últimas (m', m).

Restrição de monotonicidade: a ordem relativa das amostras no mapeamento deve ser mantida, ou seja, o mapeamento deve ser não decrescente no eixo temporal.

Restrição de continuidade: o mapeamento é feito em passos unitários, ou seja, não se pode ter saltos no eixo temporal.

A Figura 2 ilustra o mapeamento de comparação das amostras quando a distância euclidiana e a DTW são utilizadas. Note como a DTW faz uma série de relações "um para muitos" para alinhar as cristas das duas séries.

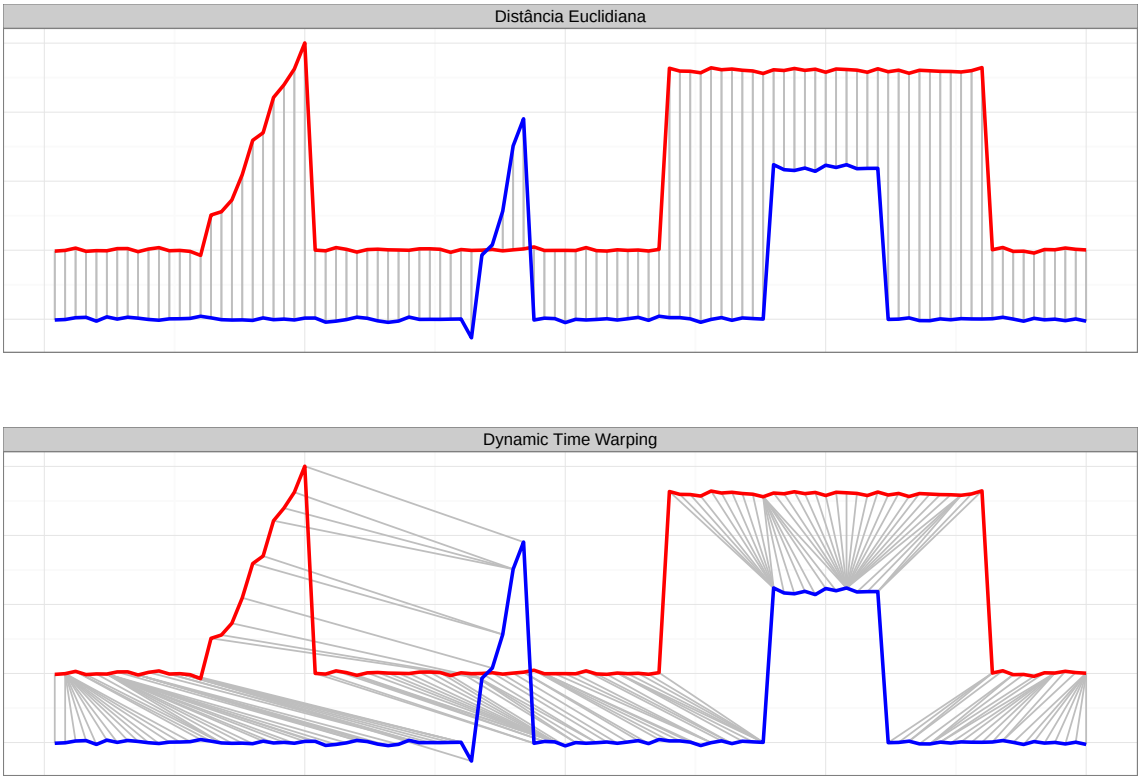

Figura 2 – Acima o alinhamento tradicional da distância euclidiana, que compara amostras tomadas no mesmo tempo, enquanto que embaixo é exibido o alinhamento proposto pela *Dynamic Time Warping*. Note como a DTW faz uma série de relações "um para muitos" para alinhar as duas cristas, e depois uma série de expansões da série azul para que o degrau seja equivalente ao degrau da série vermelha.

Usualmente a DTW é resolvida por empregar programação dinâmica. A Equação 2.4 descreve a condição inicial do algoritmo DTW.

$$dtw(i,j) = \begin{cases} 0, & \text{se } i,j = 0 \\ \infty, & \text{caso contrário.} \end{cases} \tag{2.4}$$

E a Equação 2.5 mostra a relação de recorrência do algoritmo DTW.

$$dtw(i,j) = \sqrt{(s_i - t_j)^2} + min \begin{cases} dtw(i\text{ - }1, j) \\ dtw(i, j\text{ - }1) \\ dtw(i\text{ - }1, j\text{ - }1) \end{cases} \tag{2.5}$$

No qual $i = 1 \ldots m'$ e $j = 1 \ldots m$. O valor da distância DTW é obtido por executar $dtw(m', m)$. Note que a DTW pode ser vista como uma medida de dissimilaridade própria ou como um pré-processamento que determina como as comparações das amostras devem ser feitas para então aplicar a distância euclidiana. Além disso, é importante observar que não somente a DTW retira algumas distorções no eixo temporal, como também retira o requerimento das séries

temporais terem o mesmo tamanho. Porém, ao contrário da distância euclidiana que é $O(m)$ e requer espaço constante, a DTW tem tempo de execução e requer espaço da ordem $O(m^2)$.

Por mais que a DTW apresente maior acurácia e maior robustez a distorções no eixo temporal em relação a distância euclidiana, neste trabalho a medida de dissimilaridade adotada é a distância euclidiana definida pela Equação 2.3. A principal razão para essa escolha é pelo nosso objetivo de estender e comparar com o trabalho de Hills *et al.* (2014), que utiliza a Equação 2.3. Além disso, a menos que se execute a DTW uma única vez da subsequência para a série toda, o que seria difícil justificar pois o objetivo não é distorcer a subsequência para encaixar na série total, então a distância euclidiana possui menor complexidade temporal.

2.4 Representação Simbólica

Um processo de transformação nos dados pode incorrer em perda de informação, porém acredita-se que operar sobre uma sequência de caracteres pode trazer maior robustez a ruído, além de permitir o uso de algoritmos para *strings*, como a técnica de *hashing*, logo a comunidade científica desenvolveu diversos métodos para obter uma representação simbólica de séries temporais. Como notado por Fuad e Marteau (2010), dentre os inúmeros métodos propostos para transformação dos dados, o que mais se destacou se chama *Symbolic Aggregate approXimation* (SAX) (LIN *et al.*, 2007), que tem como principais características a de garantia de equiprobabilidade de ocorrência dos símbolos utilizados na *string*, possuir uma medida de dissimilaridade que é um *lower bound* para a distância euclidiana e permitir a redução de dimensionalidade da série temporal.

2.4.1 Obtenção da Representação Simbólica

O algoritmo para obtenção da representação SAX de uma série temporal T de tamanho m necessita de dois parâmetros: α, que é o tamanho do alfabeto (que por padrão usa os caracteres a, b, c, e assim por diante); e m', que é a dimensão resultante da série.

O primeiro passo do algoritmo consiste em reduzir a dimensionaldiade da série temporal por utilizar a técnica de *Piecewise Aggregate Approximation* (PAA) (KEOGH *et al.*, 2001). A partir da série T desejamos extrair uma série $\bar{T}$ de tamanho w: $\bar{T} = \{\bar{t_1}, \bar{t_2}, \bar{t_3}, \ldots \bar{t_{w-1}}, \bar{t_{m'}}\}$. PAA obtém a série $\bar{T}$ por dividir a série T em w segmentos de igual tamanho e, de cada segmento, extrair a sua média (ver Equação 2.6).

$$\bar{t_i} = \frac{m'}{m} \times \sum_{j=\frac{m}{m'}(i-1)+1}^{\frac{m}{m'}i} t_j \tag{2.6}$$

Após a obtenção da representação PAA é preciso construir um mapeamento das médias para símbolos. É desejável que tal mapeamento produza símbolos com igual probabilidade de

ocorrência (APOSTOLICO; BOCK; LONARDI, 2003). Tal propriedade é provida pelo SAX, que pela observação empírica de que subsequências de séries *z-normalizadas* possuem distribuição Gaussiana, para ter símbolos equiprováveis basta determinar os pontos de corte que produzam a áreas de igual tamanho sob a curva Gaussiana. Os pontos de corte são uma lista ordenada de números $B = \beta_1, \ldots, \beta_{\alpha-1}$ (β_0 é para os valores de $-\infty$ até β_1 e β_α para os valores de $\beta_{\alpha-1}$ até $+\infty$), com a propriedade de que entre β_i e β_{i+1} estão $\frac{1}{\alpha}$ da área da curva Gaussiana. Essa lista de valores B com a propriedade mencionada pode ser obtida de tabelas estatísticas, como na Tabela 1, no qual é fornecido os valores de B para a entre 3 e 8.

Tabela 1 – Pontos de corte para diversos valores de α.

α	3	4	5	6	7	8
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67
β_3		0.67	0.25	0	-0.18	-0.32
β_4			0.84	0.43	0.18	0
β_5				0.97	0.57	0.32
β_6					1.07	0.67
β_7						1.15

Uma vez com a representação $\bar{T}$ e com os pontos de corte, basta para cada elemento $\bar{t}_i \in \bar{T}$ verificar a qual intervalo $(\beta_{j-1}, \beta_j]$ ele pertence, e então o valor de $\hat{t}_i$ assume o valor do j-ésimo símbolo do alfabeto de tamanho α. A Figura 3 ilustra o processo de obtenção da simbolização de uma série temporal.

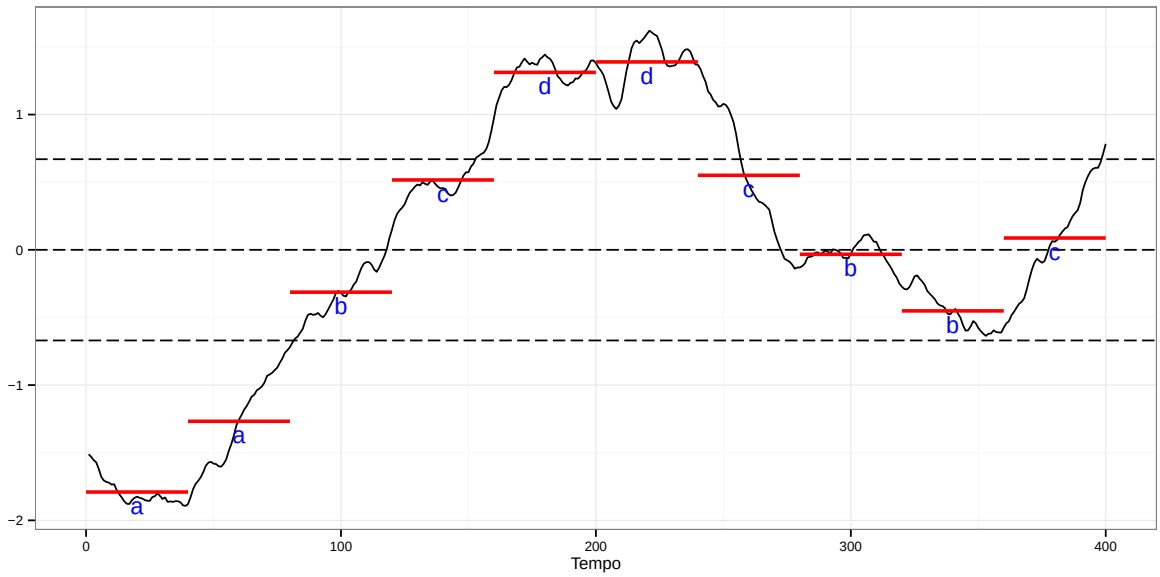

Figura 3 – Ilustração do processo de simbolização do SAX para $\alpha = 4$ e $m' = 10$ (o tracejado indica os pontos de corte B). O traçado em preto mostra a série original, enquanto que as linhas em vermelho mostram a representação dada pelo PAA e, por fim, os caracteres em azul mostram o resultado final da simbolização.

2.4.2 Medida de Dissimilaridade

Uma vez definido o processo de simbolização de uma série temporal é preciso definir uma medida de dissimilaridade. Seja a distância euclidiana como definida na Equação 2.1, então ao aplicar a transformação do PAA é possível obter um *lower bound* da distância euclidiana pela Equação 2.7.

$$dist(\bar{S},\bar{T}) = \sqrt{\frac{m}{m'}}\sqrt{\sum_{i=1}^{m'}(\bar{s}_i - \bar{t}_i)^2} \tag{2.7}$$

Então, para obter uma medida de dissimilaridade sobre símbolos basta fazer uma adaptação na Equação 2.7, resultando na Equação 2.8.

$$dist(\hat{S},\hat{T}) = \sqrt{\frac{m}{m'}}\sqrt{\sum_{i=1}^{m'} lookup(\hat{s}_i,\hat{t}_i)} \tag{2.8}$$

No qual *lookup()* é uma função que consulta uma tabela pré-calculada e retorna a distância entre os dois símbolos. A tabela a qual a função *lookup()* consulta é obtida pela Equação 2.9.

$$tabela_{linha,coluna} = \begin{cases} 0 & \text{se } |linha - coluna| \leq 1 \\ \beta_{max\{linha,coluna\}-1} - \beta_{min\{linha,coluna\}} & \text{caso contrário .} \end{cases} \tag{2.9}$$

Para o caso de exemplo da Figura 3, no qual $a = 4$, a tabela de consulta da função *lookup()* é dada pela Tabela 2.

Tabela 2 – Tabela de consulta da função *lookup()* para $a = 4$.

	a	b	c	d
a	0	0	0.67	1.34
b	0	0	0	0.67
c	0.67	0	0	0
d	1.34	0.67	0	0

Como dito anteriormente, a medida de dissimilaridade do SAX tem a propriedade de ser um *lower bound* para a distância euclidiana.

2.5 Considerações Finais

Neste capítulo foram apresentados todas as notações e definições sobre séries temporais que serão utilizadas ao longo deste trabalho. Especial atenção foi dada a representação numérica

de séries temporais, pois a menos de um algoritmo aproximado para a árvore de decisão *shapelets*, todos os algoritmos utilizam a representação numérica. Dentre as inúmeras medidas de dissimilaridade para a representação numérica nós escolhemos a da distância euclidiana para manter compatibilidade com o trabalho de Hills *et al.* (2014).

No Capítulo 3 em que serão detalhados os classificadores que serão utilizados neste trabalho as medidas de dissimilaridade aqui definidas não possuem grande destaque, pois os classificadores a serem descritos serão aplicados sob a transformada *shapelet*, logo, o principal uso da distância definida pela Equação 2.3 está em computar a transformada, que é uma matriz de distância que será definida no Capítulo 4.

CAPÍTULO

3

ALGORITMOS DE CLASSIFICAÇÃO

3.1 Considerações Iniciais

Este capítulo é dedicado aos algoritmos utilizados para classificação de séries temporais. O primeiro deles, a ser descrito na Seção 3.2, é o algoritmo *k*-vizinhos mais próximos (kNN, do inglês *Nearest Neighbors*). A importância do algoritmo kNN pode ser entendida pelo trabalho de Ding *et al.* (2008), no qual foi feita uma grande avaliação empírica sobre métodos de representação, medidas de dissimilaridade e de algoritmos de classificação de séries temporais. Dentre as conclusões, está de que embora se tenha tentando utilizar outros algoritmos de classificação, até a data de publicação, os melhores resultados publicados vieram de trabalhos que utilizaram o algoritmo kNN.

O trabalho de Ye e Keogh (2009) introduziu o que chamamos aqui de árvore de decisão *shapelets*, que de modo breve é uma árvore de decisão que opera sobre subsequências de séries temporais. Este novo classificador consegue superar a acurácia do kNN em diversos conjuntos de dados, em especial naqueles em que o ruído presente na série temporal ofusca os padrões presentes a ponto do kNN tem performance equiparável a de palpites aleatórios. O algoritmo de árvore de decisão é detalhado na Seção 3.3, porém optamos por deixar a árvore de decisão *shapelets* para o Capítulo 4, no qual são detalhados todos os trabalhos relacionados a *shapelets*.

Os outros dois algoritmos detalhados neste capítulo, o *Random Forest* na Seção 3.4 e o SVM na Seção 3.5, foram incluídos pelo seu desempenho no domínio da transformada *shapelet*. Como comentado na Seção 1.1, o trabalho de Lines *et al.* (2012) notou que algoritmos complexos, como *Rotation Forest*, *Random Forest*, *Bayesian Networks* e em especial o SVM, possuem a melhor acurácia no domínio da transformada. E no trabalho subsequente de Hills *et al.* (2014), que testou os algoritmos simples de árvore de decisão, kNN e *Naive Bayes*, e os mesmos algoritmos complexos de Lines *et al.* (2012) na transformada, foi confirmada a melhor performance do SVM seguido das *Bayesian Networks*, *Rotation Forest*, *kNN* e *Random Forest*.

Apesar de não termos incluído as *Bayesian Networks* e o *Rotation Forest* em nosso estudo, nós acreditamos que os complexos escolhidos ilustram duas maneiras únicas de classificação e servem a nossa necessidade de testar os efeitos das medidas de qualidade em ambientes distintos.

Por fim, lembramos o leitor que nenhum desses algoritmos assume como entrada uma série temporal. Para todos esses algoritmos um conjunto de dados é um conjunto de tuplas composta por um rótulo de classificação e uma sequência de atributos, que costumam ser independentes uns dos outros e não exigem qualquer ordem. Para o caso especial do algoritmo de vizinhos mais próximo é explicado como ele pode ser aplicado diretamente a séries temporais, mas tanto ele quanto os outros algoritmos aqui detalhados são aplicados sob a transformada. De qualquer forma, para facilitar o entendimento dos algoritmos, neste capítulo cada um deles será ilustrado com o uso do conjunto de dados Iris (ANDERSON, 1935; FISHER, 1936), que possui 150 tuplas de 3 tipos de plantas (setosa, versicolor e virginica), sendo que cada tupla é constituída pelos atributos de largura e de comprimento da sépala e da pétala da flor (mas para poder visualizar em 2 dimensões somente os dados da largura e do comprimento da sépala foram usados).

3.2 *k*-Vizinhos mais Próximos (kNN)

No algoritmo de vizinhos mais próximos uma tupla de m amostras é interpretada como um ponto em um espaço m dimensional. Neste espaço m dimensional é mensurada a distância entre um ponto e outro, por exemplo, pela distância euclidiana como definida na Seção 2.3.1. Então, para classificar uma nova tupla se determina as k tuplas mais próximas no conjunto de treinamento, e assim o rótulo dessa nova tupla é simplesmente o rótulo mais comum dessas k tuplas. Este é um algoritmo de fácil implementação, como mostra o Algoritmo 1, além de ter somente um único parâmetro, o valor de k.

Algoritmo 1: k-Vizinhos mais Próximos.

Entrada: Conjunto de treinamento $CD_{treinamento}$, uma nova tupla T a ser rotulada e o parâmetro k.
Saída: O rótulo c para a tupla T.

```
1 kPróximos = ∅;
2 para cada ⟨T_treinamento, c_treinamento⟩ em CD_treinamento faça
3     d = ComputaDistância(T, T_treinamento);
4     kPróximos.adiciona(⟨d, c_treinamento⟩);
5 kPróximos = selecionaMaisPróximos(kPróximos, k);
6 c = RótuloMaisComun(kPróximos);
7 retorna c
```

É possível aplicar o algoritmo de vizinhos mais próximos em séries temporais. Cada uma das m amostras (t_i) de uma série temporal T é interpretada como um atributo, e uma série temporal é interpretada como um ponto no espaço m dimensional. O ponto negativo

desta abordagem é que se perde a informação temporal contida na ordem das amostras. De qualquer forma, como notado por Ye e Keogh (2009), este classificador requer a inspeção de todo o conjunto de treinamento para classificar cada nova série temporal, e isto tem um custo computacional de $O(nm)$, limitando as suas aplicações.

Por fim, a Figura 4 mostra um exemplo de classificação pelo kNN. Neste exemplo $k = 5$ e os círculos maiores denotam o conjunto de treinamento, enquanto que os outros exemplificam uma classificação de dados não vistos.

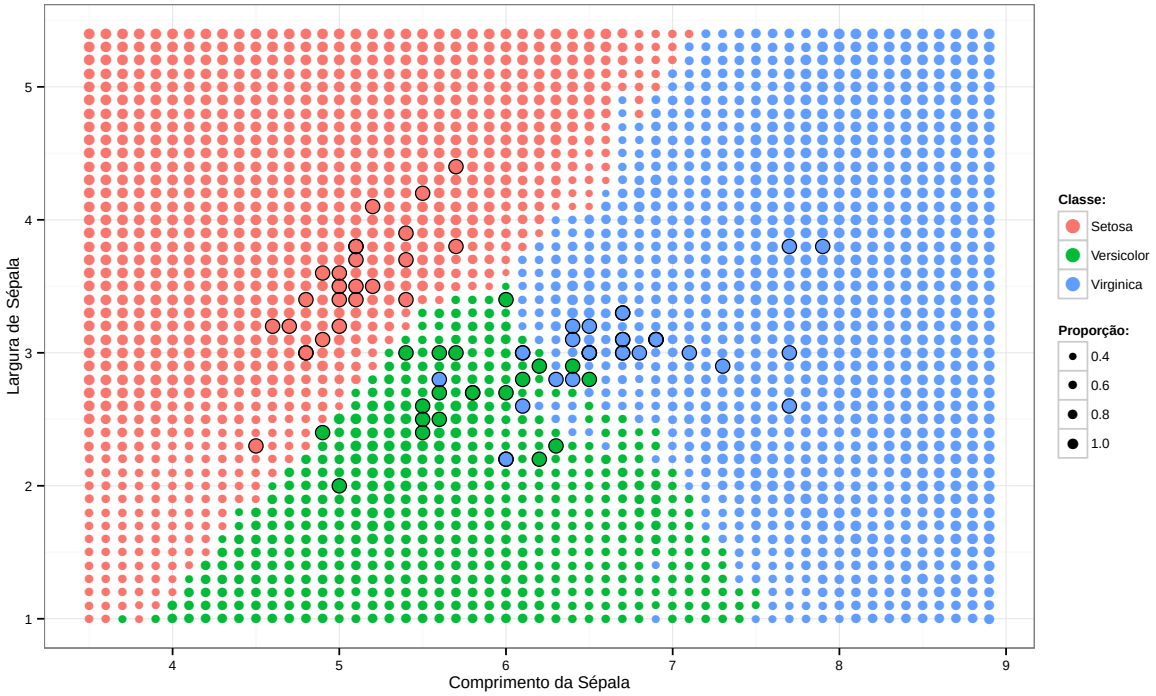

Figura 4 – Ilustração de classificação pelo algoritmo kNN ($k = 5$). As cores indicam as classes. O conjunto de treinamento é representado pelos círculos maiores, enquanto que o tamanho dos outros círculos representam a proporção da classe majoritária dentre os k mais próximos.

3.3 Árvore de Decisão

A árvore de decisão a ser detalhada é a C4.5 descrita por Quinlan (1992) para o caso em que todos todos os atributos são valores numéricos. Detalhamos a C4.5 pois ela possui o mesmo algoritmo de indução da árvore de decisão proposta por Ye e Keogh (2009).

Dado um conjunto de dados CD de n tuplas, no qual cada tupla é composta por m atributos e um rótulo de classificação c, o primeiro passo do algoritmo consiste em computar a entropia do atual conjunto pela Equação 3.1.

$$H(CD) = -\sum_{i=1}^{C} \frac{|CD_{|c_i|}|}{|CD|} log_2\left(\frac{|CD_{|c_i|}|}{|CD|}\right) \tag{3.1}$$

A seguir, para cada um dos atributos (A_i) é medido o seu ganho de informação. O ganho de informação é a diferença de entropia entre o conjunto corrente e a entropia média das partições resultantes de um ponto de corte (τ) que divide o conjunto em duas partições (ver Equação 3.2).

$$GI(CD, A_i, \tau) = H(CD) - \frac{|CD_{|\tau,A_i,menor}|}{|CD|} H(CD_{|\tau,A_i,menor}) - \frac{|CD_{|\tau,A_i,maior}|}{|CD|} H(CD_{|\tau,A_i,maior}) \quad (3.2)$$

O ponto de corte τ é um valor que particiona o conjunto corrente em duas partições, uma com as tuplas em que o atributo A_i tem valor maior que τ, e outra com as tuplas de valores A_i menores que τ. Para se chegar ao valor de τ é preciso notar que ao se ordenar as n tuplas pelo valor de A_i só existem $n-1$ partições únicas, que são as partições geradas por quaisquer valores entre dois valores sucessivos na ordenação. Usualmente usa-se o valor médio entre duas tuplas sucessivas na ordenação como valor de τ. Então se testa todos esses $n-1$ possíveis valores de τ e o maior valor de ganho de informação é armazenado para o atributo A_i. Este processo é repetido para todos os atributos, e desses atributos o de maior ganho de informação é escolhido, aqui denotado por A_{max}. A exemplificação do cálculo de ganho de informação para um ponto de corte é dado pela Figura 5.

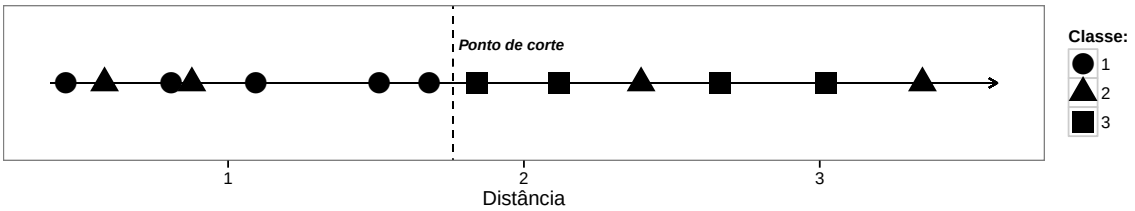

Figura 5 – Ilustração do cálculo de ganho de informação. As formas indicam as classes das tuplas, enquanto que o tracejado indica a posição de um ponto de corte ($\tau = 1.76$) após a ordenação das tuplas. Neste arranjo a entropia é $H(CD) = -\frac{5}{13}log_2(\frac{5}{13}) - \frac{4}{13}log_2(\frac{4}{13}) - \frac{4}{13}log_2(\frac{4}{13}) = 1.577$, e o ganho de informação é de $GI(CD, 2.599) = 1.577 - \frac{7}{13}0.863 - \frac{6}{13}0.918 = 0.689$.

Este atributo A_{max} é definido como a raiz de uma árvore binária e o conjunto CD é particionado de acordo com o seu τ. Então este processo é repetido recursivamente nas partições resultantes até se atingir uma condição de parada, por exemplo, tamanho mínimo de tuplas na partição do nó da árvore. Ao final deste processo, que é descrito pelo Algoritmo 2, se tem uma árvore de decisão, no qual cada nó impõe uma decisão binária de menor ou maior. Logo, para classificar uma tupla não rotulada basta percorrer a árvore de decisão de acordo com as decisões impostas pelos nós até se chegar a um nó folha, ou seja, aquele em que não há mais decisões para se tomar. Neste nó a tupla recebe o rótulo da classe majoritária da partição resultante do conjunto de treinamento naquele nó.

A Figura 6 ilustra a classificação do conjunto de dados Iris de acordo com esse algoritmo. Note como as decisões binárias se traduzem de forma gráfica como retas que cortam ortogonalmente os eixos. A adaptação deste algoritmo para operar sobre séries temporais é deixada para o Capítulo 4.

Algoritmo 2: Indução da Árvore de Decisão (para atributos numéricos).

Entrada: Conjunto de treinamento *CD*.
Saída: O nó raiz de uma árvore de decisão (*raiz*).

```
1  raiz = NULL;
2  se condiçãoDeParada(CD) = FALSO então
3      A_max = NULL;
4      para i = 1 até m faça
5          D = computaDistâncias(CD, A_i);
6          ordenaDistâncias(D);
7          A_i.τ = NULL;
8          A_i.ganho = 0;
9          para j = 1 até n − 1 faça
10             τ = (D[j]+D[j+1])/2;
11             ganho = ganhoDeInformação(CD, τ, A_i);
12             se ganho ≥ A_i.ganho então
13                 A_i.ganho = ganho;
14                 A_i.τ = τ;
15         se A_max.ganho ≤ A_i.ganho então
16             A_max = A_i;
17     raiz.menor = ÁrvoreDeDecisão(CD_{|A_max,menor});
18     raiz.maior = ÁrvoreDeDecisão(CD_{|A_max,maior});
19 senão
20     raiz.rótulo = RótuloMaisComum(CD);
21 retorna raiz
```

Dentre as vantagens deste algoritmo estão o fato de induzir um modelo interpretável, afinal, a classificação é um conjunto de regras binárias. Além disso, apesar do custo da indução da árvore de decisão, uma vez que ela esteja pronta a classificação tem custo computacional linear na profundidade da árvore multiplicada pelo custo de tomar a decisão e, ao contrário do kNN, não requer manter todo o conjunto de treinamento.

3.4 *Random Forest*

Todos os algoritmos supervisionado de aprendizado de máquina enfrentam o dilema do balanço entre bias e variância. Os algoritmos buscam definir uma função que faz uma ligação entre os atributos que caracterizam uma tupla com um valor de rótulo, porém esta relação possui um ruído natural que causa um erro irredutível. Algoritmos de grande bias superestimam este erro irredutível ao elaborarem modelos com suposições simplistas dos dados vistos, causando grande erro em dados não vistos. Enquanto que algoritmos de alta variância elaboram modelos de grande complexidade que se atêm fortemente aos dados vistos, assim capturando o ruído nos dados vistos mas não generalizando bem a dados não vistos.

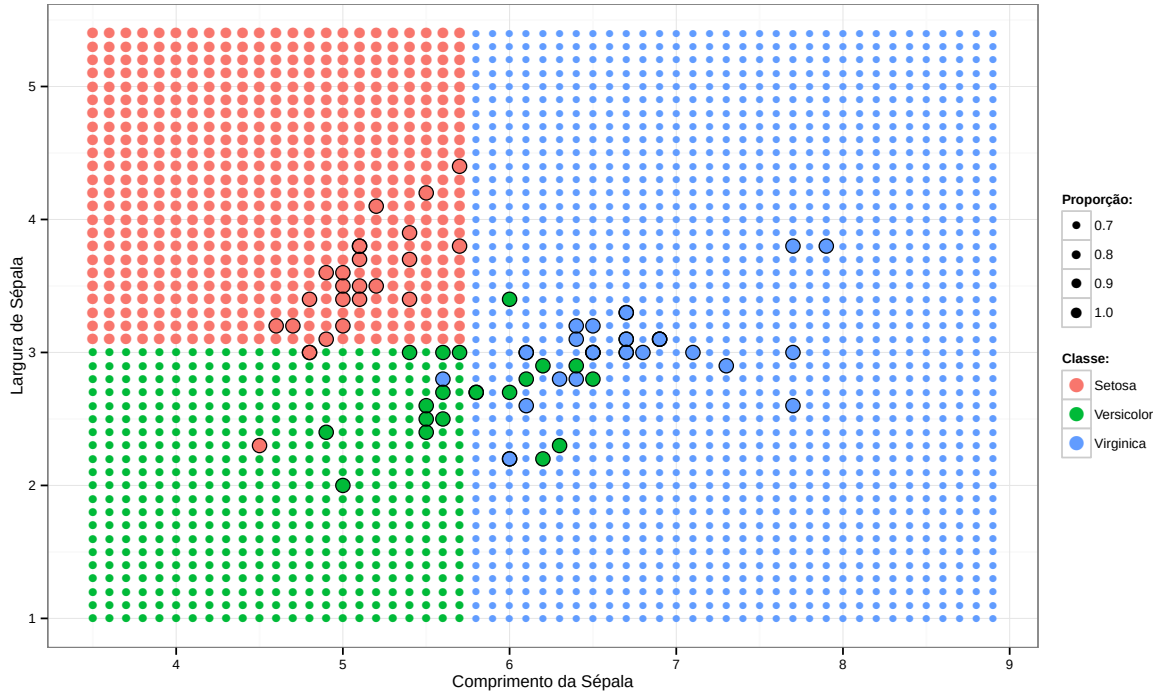

Figura 6 – Ilustração de classificação pelo algoritmo de árvore de decisão. As cores indicam as classes. O conjunto de treinamento é representado pelos círculos maiores, enquanto que o tamanho dos outros círculos representam a proporção da classe majoritária da partição resultante do conjunto de treinamento que levou a determinada classificação.

Árvores de decisão são capazes de capturar padrões complexos nos dados, e caso tenham grande profundidade elas possuem baixo *bias*. Como contrapartida a sua capacidade de capturar padrões complexos, árvores de decisão costumam também incorporar o ruído dos dados (FRIEDMAN; HASTIE; TIBSHIRANI, 2009). O classificador *Random Forest* (BREIMAN, 2001) consegue ter baixo bias e baixa variância por duas razões:

1. Para reduzir a alta variância das árvores de decisão inúmeras árvores de decisão são induzidas, e a classificação se dá pelo voto de maioria das árvores, diluindo assim a variância.

2. Como a árvore de decisão descrita é um algoritmo deterministico, é preciso que haja diferenças nos dados para gerar árvores diferentes.

   - A primeira diferença é obtida por induzir árvores a partir de um conjunto de dados que é obtido por uma amostragem com reposição das tuplas, ou seja, *bootstrap sampling*.
   - A segunda diferença é obtida por uma modificação no ato de computar o ganho de informação. Antes de computar o ganho de informação de todos os atributos para escolher o de máximo para compor o novo nó da árvore, é feita uma amostragem aleatória dos atributos, ou seja, não são todos os atributos que são avaliados.

Sendo assim, o *Random Forest* é sumariamente um grande conjunto de árvores de decisão não-correlacionadas. Na prática ele é um algoritmo de alta acurácia (FRIEDMAN; HASTIE; TIBSHIRANI, 2009), no qual ao contrário das fronteiras de decisão de uma árvore de decisão comum que faz cortes ortogonais aos eixos, graças a sua floresta e o sistema de votação, as suas fronteiras de decisão são maleáveis, como mostra a Figura 7. Por fim, este algoritmo tem dois parâmetros: a quantidade de árvores que irão compor a floresta, no qual em um estudo empírico foi mostrado que ter mais de 100 árvores traz pouco benefício na acurácia (OSHIRO; PEREZ; BARANAUSKAS, 2012); e a quantidade de atributos amostrados para adição de um novo nó na árvore, que costuma ser $\sqrt{m}$, no qual m é a quantidade total de atributos.

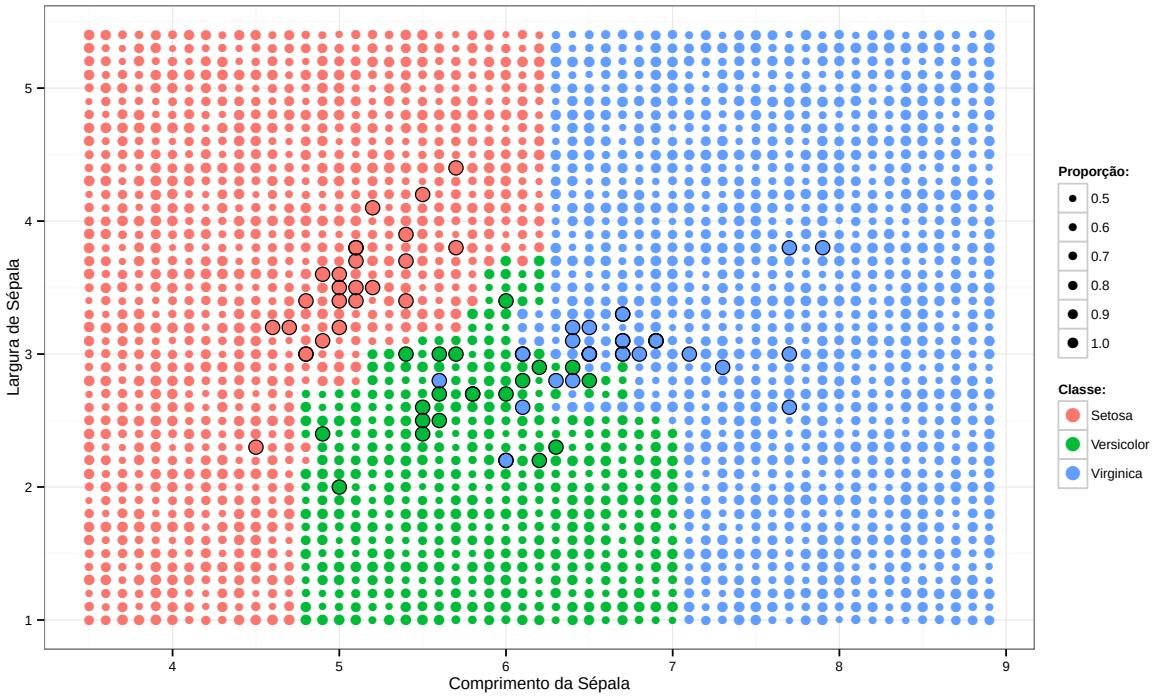

Figura 7 – Ilustração de classificação pelo algoritmo *Random Forest*. As cores indicam as classes. O conjunto de treinamento é representado pelos círculos maiores, enquanto que o tamanho dos outros círculos representam a proporção da classe majoritária nos votos das árvores.

3.5 *Support Vector Machines* (SVM)

Sumariamente, o classificador *Support Vector Machines* (SVM) (VAPNIK, 1996) busca encontrar o hyperplano ótimo que separa as tuplas de diferentes classes ao mesmo tempo em que maximiza a distância do plano aos pontos mais próximos, sendo que tal distância é denominada margem, e os pontos que estão no limite da margem são denominados *support vectors*, como ilustra a Figura 8. A classificação se dá por medir a distância da tupla ao plano e avaliar pelo sinal da distância em qual lado ela está.

O problema ocorre quando as classes não são linearmente separáveis e o algoritmo tem que ser capaz de fazer um balanço entre maximizar a margem enquanto tolera erros na

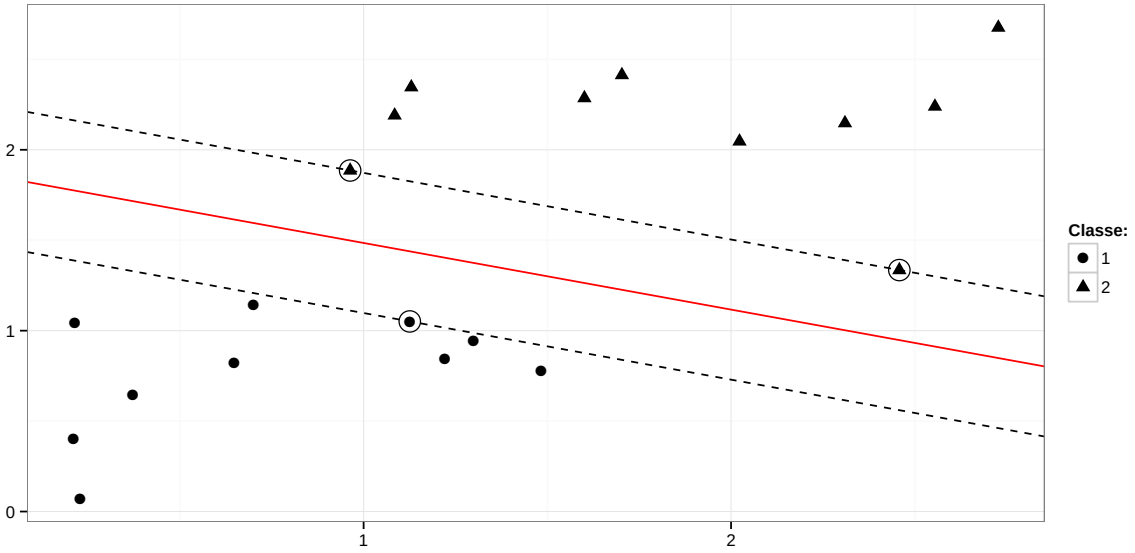

Figura 8 – A linha central ilustra o plano separador que maximiza as margens (linhas tracejadas). Os pontos circulados são os *support vectors*, que são os pontos que determinaram o plano e as margens.

classificação. A seguir será feita uma breve descrição sobre como o algoritmo determina tal plano, para uma explicação completa recomenda-se a leitura de Friedman, Hastie e Tibshirani (2009).

Seja um conjunto de dados de n tuplas, cada qual composta por um vetor de atributos x_i e um rótulo de classificação y_i, no qual por simplicidade temos somente duas classes, $y_i \in \{-1, 1\}$. O plano separador é definido pela Equação 3.3.

$$\{x : f(x) = x^T \beta + \beta_0 = 0\} \tag{3.3}$$

No qual β é um vetor unitário ($||\beta|| = 1$). E como dito anteriormente a classificação é dada pelo sinal da distância de um ponto ao plano: $Classe(x_i) = sinal[x_i^T \beta + \beta_0]$.

No problema original em que as classes são linearmente separáveis maximizar o tamanho da margem M é o mesmo que minimizar $||\beta||$, pois $M = \frac{1}{||\beta||}$, e o plano ideal é obtido por resolver este problema de otimização da Equação 3.4.

$$\begin{aligned} &\underset{\beta,\beta_0}{\text{minimizar}} && ||\beta|| \\ &\text{sujeito a} && y_i(x_i^T \beta + \beta_0) \geq 1,\ i = 1, \ldots, n. \end{aligned} \tag{3.4}$$

Quando as classes não são linearmente separáveis se é adicionado a variável de controle ξ_i para manter a inequação e permitir pontos dentro ou no lado oposto da margem. Assim a

Equação 3.4 se torna 3.5.

$$\begin{aligned} \underset{\beta,\beta_0}{\text{minimizar}} \quad & ||\beta|| \\ \text{sujeito a} \quad & y_i(x_i^T\beta+\beta_0) \geq 1-\xi_i,\ \forall_i, \\ & \xi_i \geq 0, \sum \xi_i \leq C \text{ (constante)} \end{aligned} \tag{3.5}$$

A Equação 3.5 é um problema de otimização convexo e pode ser resolvido por utilizar os multiplicadores de Lagrange. No entanto, para resolver a Equação 3.5 computacionalmente é mais interessante reescreve-la como na Equação 3.6.

$$\begin{aligned} \underset{\beta,\beta_0}{\text{minimizar}} \quad & \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{sujeito a} \quad & \xi_i \geq 0,\ y_i(x_i^T\beta+\beta_0) \geq 1. \end{aligned} \tag{3.6}$$

No qual C é uma constante que penaliza a presença de pontos dentro da margem ou do lado oposto. A função primal de Lagrange da Equação 3.6 é a Equação 3.7.

$$L_P = \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(x_i^T\beta+\beta_0)-(1-\xi_i)] - \sum_{i=1}^{m}\mu_i\xi_i \tag{3.7}$$

Em que desejamos minimizá-la em respeito a β, β_0 e ξ_i. Para isso derivamos os parâmetros da Equação 3.7 e igualamos a zero, obtendo as Equações 3.8, 3.9 e 3.10.

$$\beta = \sum_{i=1}^{n}\alpha_i y_i x_i, \tag{3.8}$$

$$0 = \sum_{i=1}^{n}\alpha_i y_i, \tag{3.9}$$

$$\alpha_i = C - \mu_i,\ \forall_i \tag{3.10}$$

Ao se utilizar esses resultados na Equação 3.7 se obtém a Equação 3.11 a ser otimizada (sujeita a $0 \leq \alpha_i \leq C$ e $\sum_{i=1}^{m}\alpha_i y_i = 0$).

$$L_D = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j \tag{3.11}$$

Com isso temos um método para encontrar o plano da Figura 8 que é capaz de tolerar alguns erros de classificação de acordo com uma constante C.

Por fim, o classificador SVM emprega mais uma técnica para conseguir encontrar o plano ideal, que é a ideia do uso de *kernel* para procurar por um plano em um espaço expandido de atributos. Como notado, temos originalmente m atributos, porém iremos expandir esse conjunto

para um tamanho de p ($p > m$). Denotaremos por $h(x_i)$ a representação de x_i neste espaço expandido, e denotaremos por $\langle h(x_i), h(x_j)\rangle$ o produto escalar entre $h(x_i)$ e $h(x_j)$. Sendo assim, a Equação 3.11 pode ser re-escrita como na Equação 3.12.

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle h(x_i), h(x_j)\rangle \tag{3.12}$$

E por substituir a Equação 3.8 em 3.3 obtemos a Equação 3.13.

$$f(x_j) \quad = \sum_{i=1}^{n} \alpha_i y_i \langle h(x_i), h(x_j)\rangle + \beta_0 \tag{3.13}$$

E por fim definimos o *kernel* como sendo a função que retorna este produto escalar, Equação 3.14.

$$K(x_i, x_j) = \langle h(x_i), h(x_j)\rangle \tag{3.14}$$

O interessante é que dependendo de como se deseja expandir o conjunto de atributos para computar o produto escalar não é preciso obter a representação de x_i em $h(x_i)$. Por exemplo, seja x_i composto por dois atributos (A_1, A_2), e desejamos fazer uma expansão polinomial de grau 2, ou seja, desejamos obter o seguinte conjunto de atributos: $A_1A_1, A_1A_2, A_2A_1, A_2A_2$. Então para este caso faça $K(x_i, x_j) = (\langle x_i, x_j\rangle)^2$. Para ilustrar, seja $x_i = (1,2)$ e $x_j = (3,4)$, então $h(x_i) = (1,2,2,4)$ e $h(x_j) = (9,12,12,16)$, logo $\langle h(x_i), h(x_j)\rangle = 9+24+24+64 = 121$. Podemos chegar ao mesmo resultado utilizando o *kernel* sem precisar representar os dados neste espaço de maior dimensão: $K(x_i, x_j) = (1 \times 3 + 2 \times 4)^2 = 121$.

A razão para se expandir a dimensão é que nesse espaço dimensional os dados podem ser linearmente separáveis. No entanto, um plano neste espaço dimensional maior ao ser projetado no espaço original pode retornar com diversas curvas, que é o que intuitivamente dá o seu alto poder de classificação. De fato, a variável C quando possui valor baixo desencoraja valores que infrinjam a margem, resultando em fronteiras de muitas curvas. Logo, além do parâmetro C, outro parâmetro de suma importância é o tipo de *kernel* a ser utilizado. A Figura 9 ilustra a classificação do conjunto de dados Iris pelo uso de um *kernel* linear.

3.6 Considerações Finais

Neste capítulo foram apresentados os algoritmos de 4 classificadores. O algoritmo da árvore de decisão merece destaque, pois é assumida sua compreensão ao introduzirmos a árvore de decisão *shapelets* no Capítulo 4, que de modo breve é uma árvore de decisão no qual os atributos são subsequências. No Capítulo 4 também é definido o que é um *shapelet* e como obter a transformada *shapelet*, que por alto é uma matriz de distância entre subsequências e séries

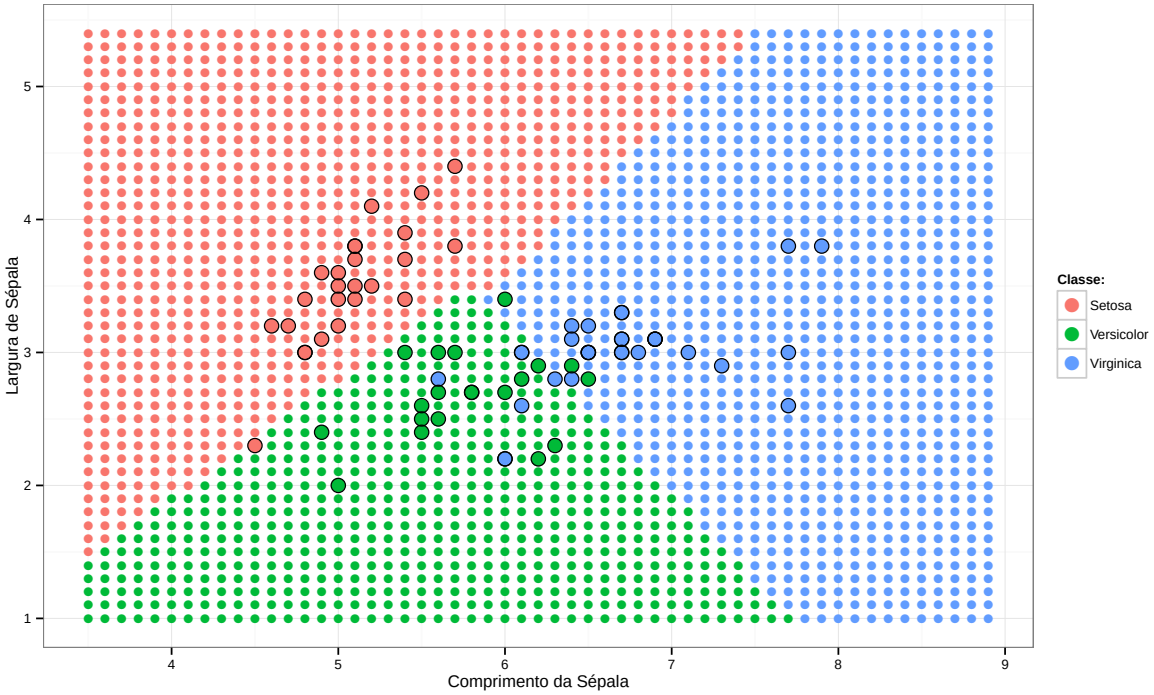

Figura 9 – Ilustração de classificação pelo algoritmo SVM com *kernel* linear e $C = 1$. As cores indicam as classes. O conjunto de treinamento é representado pelos círculos maiores.

temporais. Será sob essa transformada que iremos realizar os nossos experimentos que avaliam como a acurácia dos classificadores aqui descritos é afetada em diversas condições.

A escolha dos outros algoritmos se deu pelo seu uso ou poder de classificação demonstrado em trabalhos passados, e pela forma como a classificação é feita. Para cada um dos classificadores aqui descritos foi exibida uma ilustração de suas fronteiras de classificação, pois um de nossos objetivos é avaliar como a escolha de atributos para a transformada afeta a acurácia. Classificadores que exibem fronteiras de classificação distintas estressam os atributos em diferentes formas, evidenciando qualquer problema que o modo de escolha dos atributos possa ter.

CAPÍTULO

4

SHAPELETS

4.1 Considerações Iniciais

Este capítulo trata sobre *shapelets*, que são subsequências escolhidas para diferenciar séries de classes distintas. Esta primitiva foi introduzida no trabalho de Ye e Keogh (2009), o qual também apresentou a árvore de decisão *shapelets*, que é uma árvore de decisão que embute os *shapelets* em seus nós, como é descrito na Seção 4.2.

Na Seção 4.3, sobre a transformada *shapelet*, os *shapelets* sofrem uma mudança de paradigma, pois deixam de ser meras subsequências de um nó de uma árvore de decisão para serem os atributos de uma matriz de distância entre as séries temporais e os atributos, esta matriz de distância é o resultado da transformada. No entanto, para obter esta transformada é preciso escolher os *shapelets* que irão compor a transformada, e para isso são utilizadas medidas de qualidade (Seção 4.3.1). Uma vez obtida a transformada é possível aplicar diversos classificadores sem que eles tenham que ser adaptados para séries temporais, pois este processo de obtenção da transformada dissocia o processo de manipulação das séries temporais da classificação das mesmas. Além disso, este capítulo também contém a descrição do estado da arte para classificação de séries pela transformada (Seção 4.3.3), incluindo a sua abordagem para redução do espaço de busca dos *shapelets*.

4.2 Árvore de Decisão *Shapelets*

Shapelets são subsequências de séries temporais que, de certa forma, são as mais representativas de sua classe, e a árvore de decisão *shapelets* os embute em seus nós para obter modelos de boa acurácia e interpretabilidade (YE; KEOGH, 2009).

O primeiro passo para induzir uma árvore de decisão é medir a entropia do conjunto de dados, como descrito na Seção 3.3. Após o calculo da entropia é preciso definir os atributos

para o cálculo do ganho de informação. Na árvore de decisão *shapelets* os atributos são todas as possíveis subsequências do conjunto de dados *CD*, sendo que um conjunto de dados de n séries temporais, cada qual com tamanho m, tem $n\frac{m(m+1)}{2}$ subsequências. Todas essas subsequências são as candidatas a *shapelet*, que é a subsequência de maior ganho de informação. Para calcular o ganho de informação é preciso computar a distância de uma subsequência à todas séries do conjunto de dados, para isso se utiliza a distância euclidiana adaptada da Equação 2.3, que é capaz de computar a distância entre séries temporais de tamanhos diferentes. O resto do processo, inclusive de classificação de uma nova série, é como descrito na Seção 3.3.

É dito que a subsequência *shapelet* é representativa de sua classe, pois espera-se que séries da mesma classe que a sua contenham uma subsequência de grande similaridade à ela e, ao mesmo tempo, as séries de outras classes não possuam uma subsequência com tal similaridade. Se isso não fosse verdadeiro, então a ordenação das séries (ilustrado pela Figura 5) não traria um particionamento bom, logo, ela deve ser representativa de sua classe.

Para ilustrar o poder de classificação da árvore de decisão *shapelets* é mostrado um exemplo com o conjunto de dados TwoLeadECG, disponível no repositório da UCR (CHEN *et al.*, 2015). Este conjunto de dados tem somente duas classes, e o conjunto de treinamento é composto por 23 séries temporais, enquanto que o conjunto de teste possui 1139 séries, todas de 82 amostras. Uma árvore de decisão com somente um nó, ou seja, uma única subsequência, é capaz de classificar este conjunto de dados com acurácia de 92%, como mostra a Figura 10. Mais do que isso, este exemplo também mostra o poder de interpretabilidade desta técnica, pois ao inspecionar o *shapelet* é possível entender que o que separa as classes é a crista que ocorre próxima a 30ª amostra das séries da classe 1.

Além da técnica de extrair todas as subsequências prover um meio de identificar padrões locais e evitar que o ruído presente ao longo de toda a série possa degradar o poder de classificação, ao se procurar pela ausência ou presença de determinado padrão por deslizar a subsequência ao longo da série temporal se ganha em robustez, pois não é mais exigido que o padrão apareça na mesma região em todas as séries. Como contrapartida a esses benefícios este algoritmo tem um alto custo computacional. Existem cerca de $O(nm^2)$ subsequências, e para computar a distância de uma delas à todas as séries é preciso $O(nm^2)$, logo o algoritmo é $O(n^2m^4)$ (sem levar em conta o tempo para computar o ganho de informação). Ye e Keogh (2009) estão cientes deste problema e recomendam o uso de duas técnicas para diminuir o tempo de indução da árvore:

abandonamento precoce: durante o processo de computar a distância euclidiana de uma subsequência à uma série temporal se desliza a subsequência e se calcula inúmeras distâncias euclidianas para utilizar somente a menor. É possível utilizar a menor distância encontrada até o momento para abandonar o calculo da distância euclidiana corrente caso a distância corrente se torne maior do que a menor encontrada.

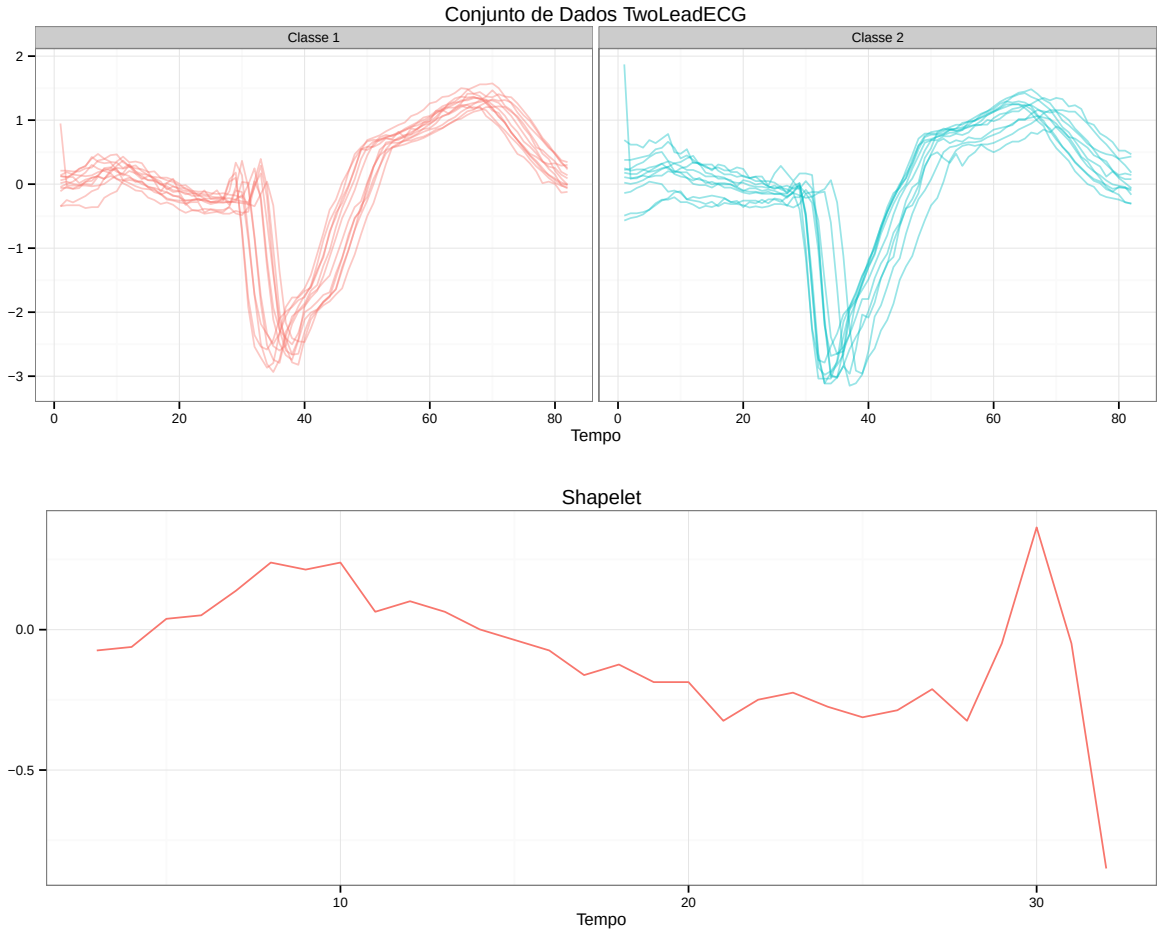

Figura 10 – Exemplo de classificação pela árvore de decisão *shapelets*. Ao topo é mostrado o conjunto de dados TwoLeadECG, disponível no repositório da UCR (CHEN *et al.*, 2015). Embaixo é mostrada a subsequência de maior ganho de informação, ou seja, o *shapelet*. Uma árvore de decisão com somente este *shapelet* é capaz de obter uma acurácia de 92%. Note como ele representa a crista que ocorre próxima a 30º amostra das séries da classe 1.

poda no cálculo do ganho de informação: como se deseja somente a subsequência de maior ganho de informação, salve o valor do maior ganho de informação encontrado até o momento. Então para as subsequências seguintes comece de forma otimista, ou seja, assumindo que todas as séries de classes distintas estão divididas perfeitamente na ordenação, e então compute a distância da subsequência à uma série de cada classe por vez, até que a ordenação fique tão embaralhada que a entropia não permita obter um ganho de informação melhor do que o valor salvo, podendo assim abandonar este candidato a *shapelet*. Este processo é ilustrado pela Figura 11.

Apesar dessas duas técnicas reduzirem o tempo de indução da árvore, elas não mudam a complexidade de $O(n^2m^4)$ do algoritmo.

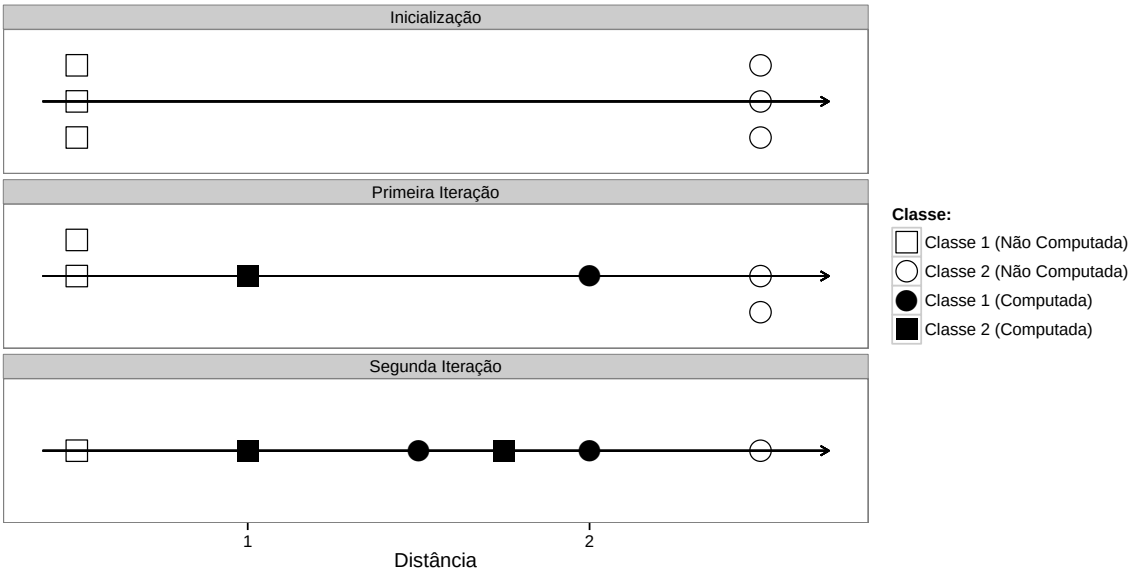

Figura 11 – Ilustração da poda no cálculo do ganho de informação. Inicialmente se assume a ordenação mais otimista das séries temporais, aquela em que as séries de diferentes classes estão separadas, maximizando o ganho de informação. Então em cada iteração subsequente se computa a real distância de uma série temporal de cada classe à subsequência. Entre cada iteração se calcula o ganho de informação possível. Caso o ganho de informação não possa ser melhor do que o encontrado em uma outra subsequência qualquer, então o cálculo do ganho de informação da atual subsequência pode ser interrompido. Caso contrário se prossegue até o cálculo de todas as distâncias, com uma possível atualização do melhor ganho de informação encontrado.

4.2.1 Trabalhos Relacionados

O algoritmo exato estado da arte para a árvore de decisão *shapelets* é dado por Mueen, Keogh e Young (2011), que possui complexidade $O(n^2m^3)$. Tal complexidade é atingida por duas técnicas de otimização: a primeira delas explora a desigualdade triangular da distância euclidiana entre os candidatos a *shapelet* para avaliar se o atual candidato a *shapelet* pode ter um ganho de informação melhor do que o encontrado até o momento; a segunda delas envolve o uso de espaço extra ($O(nm^2)$) para memorizar uma série de somatórias para tornar mais rápido o processo de *z-normalização* e de avaliação de dissimilaridade (MUEEN; NATH; LIU, 2010; MUEEN, 2013).

Para o caso de algoritmo aproximado, ou seja, quando não garante encontrar o melhor *shapelet*, existem dois algoritmos notáveis. O trabalho de Gordon, Hendler e Rokach (2012) tem como objetivo avaliar maneiras de se explorar o espaço de *shapelets* possíveis para encontrar um modelo de grande acurácia de forma rápida, e chegaram a conclusão de que explorar de forma aleatória provém bons resultados, até mesmo quando poucos candidatos a *shapelets* são testados.

O outro algoritmo, chamado *Fast Shapelets*, foi introduzido em por Rakthanmanon e Keogh (2013) e faz uso da representação simbólica SAX explicada na Seção 2.4. O primeiro passo do algoritmo consiste em extrair todas as subsequências de um determinado tamanho e obter a representação simbólica das mesmas. Então o algoritmo faz um processo chamado de

projeções aleatórias para descobrir quais são as subsequências comuns, ou seja, aquelas que tem maior chance de serem boas candidatas a *shapelet*. Sejam duas subsequências na representação SAX, $S = abbcd$ e $T = abccd$, no qual a menos da terceira amostra elas são idênticas, mas esta diferença pode ter sido causada por um pequeno valor acima do limite para o símbolo 'b' e deveriam ser tratadas como idênticas. Para resolver esse problema, e mitigar falsos negativos, se escolhe aleatoriamente um conjunto de amostras que não serão comparadas, por exemplo, a 3ª amostra, e com isso se faz a projeção de todos os candidatos a *shapelets*. Na projeção cada subsequência que se iguala com outra tem o seu contador incrementado. As subsequências de grande contagem são inspecionadas no passo seguinte, que consiste em averiguar a contagem de colisões por classe. A partir da contagem de colisão por classe se computa um *rank*, que dá maior valor a subsequências que só se igualam com outras subsequências de mesma classe. As de k maior *rank* são selecionadas para o próximo passo, no qual elas são retornadas a representação numérica original e seus ganhos de informação são computados como na árvore de decisão *shapelet* original (Seção 4.2). A de maior ganho de informação, depois de executar este processo para todos os tamanhos de subsequência, é utilizada como o *shapelet*.

Este algoritmo descrito tem um processo complexo, mas que os autores justificam por ter um ganho de cerca de 3 ordens de magnitude no tempo de execução em relação ao estado da arte de Mueen, Keogh e Young (2011). No entanto, em nossa opinião, seria interessante um comparativo com o algoritmo de simples amostragem aleatória de Gordon, Hendler e Rokach (2012), pois talvez essa complexidade não seja necessária.

4.3 Transformada *Shapelet*

A partir desta seção *shapelet* não será mais somente a subsequência de maior ganho de informação, mas sim qualquer subsequência que tenha uma medida de qualidade associada, ou seja, todos os candidatos a *shapelets* passam a ser um *shapelet*. Esta re-definição se faz necessária pois na transformada *shapelet* uma série temporal é descrita por diversos atributos, sendo que cada atributo é um *shapelet*. Como utilizar uma grande quantidade de *shapelets* pode causar *overfitting*, é preciso escolher alguns dentre todos os possíveis, por isso se faz uso de medidas de qualidade de *shapelets* que indicam a sua utilidade.

A Seção 4.3.1 descreve como computar uma medida de qualidade a partir de um *shapelet* e descreve as três medidas de qualidades utilizadas nesse trabalho, incluindo a nossa proposta, denominada *in-class transitions*. Uma vez com um conjunto de *shapelets* é possível obter a transformada de um conjunto de dados, que é uma matriz de distância entre as séries temporais e os *shapelets*, como é descrito na Seção 4.3.2.

Na seção seguinte, 4.3.3, são detalhados todos os passos do modelo estado da arte para classificação de séries temporais, incluindo o seu método de redução de tempo de execução por explorar somente *shapelets* de um determinado o tamanho, o que será alvo de uma detalhada

avaliação no Capítulo 5. Por fim, na Seção 4.3.4 é feita uma discussão sobre os trabalhos relacionados e as limitações da transformada.

4.3.1 Medidas de Qualidade de Shapelets

Para obter a transformada *shapelet* a partir de um conjunto de dados *CD* é preciso obter um conjunto de *shapelets* que serão os atributos que caracterizarão as séries temporais na transformada. Esta seção é dedicada a como obter tal conjunto de atributos.

Em linhas gerais o processo consiste em executar os mesmos passos da árvore de decisão, mas somente para o primeiro nível, ou seja, sem particionar o conjunto de dados a medida que novos nós são adicionados a árvore. Além disso, como neste trabalho são exploradas diversas medidas de qualidade, que serão detalhadas a seguir, estas outras medidas são computadas em conjunto com a de ganho de informação. O Algoritmo 3 formaliza o processo de extração e computação das medidas de qualidade para os *shapelets*, que começa por gerar todas as p subsequências possíveis e então, para cada uma delas, computar suas respectivas medidas de qualidade a partir da distância da subsequência para todas às séries temporais.

Algoritmo 3: Extração e computação das medidas de qualidade dos *shapelets*.

Entrada: Conjunto de dados *CD* e os parâmetros *min* e *max* que indicam o intervalo de tamanho das subsequências a serem geradas.
Saída: Conjunto de *shapelets* $Sh = \{\langle S_1, q_1\rangle, \ldots, \langle S_p, q_p\rangle\}$, no qual S_i descreve a subsequência que caracteriza o *shapelet* e q_i são as medidas de qualidade associadas ao *shapelet*.

```
1 Sh = ∅;
2 subsequências = GeraTodasAsSubsequências(CD, min, max);
3 para cada subsequência S em subsequências faça
4     D_S = ComputaDistâncias(S, CD);
5     q = ComputaMedidasDeQualidade(D_S);
6     Sh.adiciona(⟨S, q⟩);
7 retorna Sh
```

Neste trabalho são exploradas algumas medidas de qualidade alternativas ao ganho de informação, e a razão para isso se deve ao fato de que em uma árvore de decisão é natural pensar no uso do ganho de informação, dado que ele propõe um ponto de corte binário, mas é questionável que o ganho de informação consiga capturar toda a informação oferecida pela subsequência para problemas de múltiplas classes (que exige mais de um ponto de corte). As medidas de qualidade por nós exploradas são: ganho de informação, pois até a introdução da transformada era a única medida utilizada, e mesmo após o trabalho de Hills *et al.* (2014), na transformada *shapelet* não houve nenhum trabalho de avaliação extensiva com outra medida de qualidade; *f-statistic*, pois segundo o trabalho de Hills *et al.* (2014) para a árvore de decisão *shapelets* ela costuma ter a melhor acurácia (embora sem diferença estatística) e o menor tempo de execução, sendo o seu uso na transformada *shapelet* uma simples sugestão; e a

in-class transitions, que é a medida de qualidade proposta nesta dissertação. Todos elas são computadas a partir do conjunto de distâncias da subsequência S às séries temporais de CD (valores armazenados em D_S):

ganho de informação: é computada da mesma maneira como definido no algoritmo de árvore de decisão (ver Seção 3.3).

***f-statistic*:** é a mesma métrica utilizada na ANOVA (*Analysis of Variance*), afinal, ela arbitra a hipótese nula no teste estatístico de que todas as médias das classes possuem o mesmo valor ($\bar{D_{S|c_i}}$, no qual c_i é uma das C classes). Seu calculo é dado pela Equação 4.1.

$$FStatistic(D_S) = \frac{\sum_{i=1}^{C}(\bar{D_{S|c_i}} - \bar{D_S})^2/(C-1)}{\sum_{i=1}^{C}\sum_{d_j \in D_{S|c_i}}(d_j - \bar{D_{S|c_i}})^2/(n-C)} \tag{4.1}$$

***in-class transitions*:** tem como inspiração a medida de qualidade ganho de informação e é de fácil computação. Após ordenar os elementos pela distância, bem como é feito no ganho de informação, basta contar quantos elementos sucessivos na ordenação são da mesma classe. Por exemplo, na Figura 5 de ilustração do ponto de corte o valor desta medida seria de 4, enquanto que na Figura 12, que tem os mesmos elementos da figura mas em uma diferente configuração, o valor da medida seria de 8. Esta medida é uma crítica ao fato de que a entropia não leva em consideração a ordenação dos elementos, pois tanto a Figura 5 quanto a Figura 12 possuem a mesma entropia, e ao utilizar o mesmo ponto de corte na duas figuras elas teriam o mesmo ganho de informação, mas intuitivamente é mais fácil separar e classificar os elementos da Figura 12.

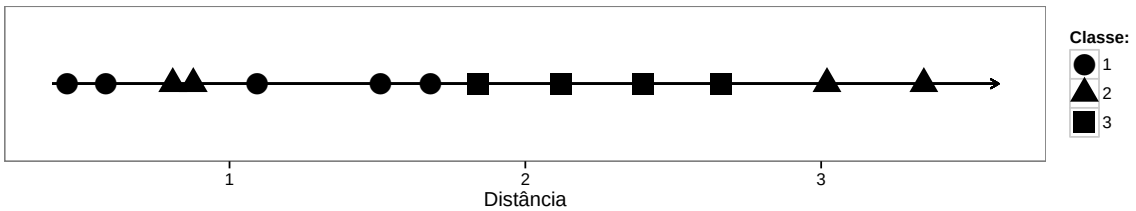

Figura 12 – Ilustração da medida de qualidade *in-class-transitions*. Seu valor é simplesmente a contagem de quantos elementos sucessivos na ordenação são da mesma classe. Neste exemplo a medida de qualidade tem o valor de 8.

Todas as medidas de qualidade possuem um tempo linear para computação, porém tanto o ganho de informação quanto a *in-class transitions* requerem que os elementos estejam ordenados, o que geralmente tem um custo de $O(nlogn)$ de ordenação. De toda maneira, para todas elas quanto maior for o seu valor melhor.

Na Seção 4.3.3, que trata sobre o estado da arte, será detalhado a escolha desses k *shapelets*, por ora assuma que as de k maiores valores são escolhidas.

4.3.2 A transformada

Um vez com o conjunto de dados *CD* e um conjunto de *shapelets Sh* é possível obter a transformada *shapelet*. A transformada *shapelet* é uma matriz numérica, no qual cada posição é um valor numérico que é obtido por medir a distância de um *shapelet* à uma série temporal (pela Equação 2.3), além de incluir uma coluna para identificar a classe da série temporal para ser utilizada pelo classificador. A esquematização de tal transformada *shapelet* é dada pela Tabela 3.

Tabela 3 – Esquematização da transformada *shapelet*. Seja um conjunto de dados $CD = \{\langle T_1, c_1\rangle, \ldots, \langle T_n, c_n\rangle\}$ e um conjunto de k *shapelets* ($Sh = \{\langle S_1, q_1\rangle, \ldots, \langle S_k, q_k\rangle\}$), então a transformada é obtida por mensurar a distância de cada uma das séries temporais T_i à cada um dos k *shapelets* pela Equação 2.3, logo $a_{ij} \in \mathbb{R}$.

Série Temporal	S_1	S_2	$\ldots$	S_k	Classe
T_1	a_{11}	a_{12}	$\ldots$	a_{1k}	c_1
T_2	a_{21}	a_{22}	$\ldots$	a_{2k}	c_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
T_n	a_{n1}	a_{n2}	$\ldots$	a_{nk}	c_n

A consequência direta de se obter esta transformada é que se dissocia o processo de descoberta e extração de subsequências do processo de classificação, pois uma vez que se transforma o problema em uma matrix numérica com uma coluna de rótulos se pode utilizar qualquer classificador, como o *Random Forest* e o SVM que foram descritos no Capítulo 3. É desejável este processo pois o trabalho de Hills *et al.* (2014) mostrou que eles oferecem uma maior acurácia do que embutir o processo de descoberta e extração de subsequências na árvore de decisão.

Na próxima Seção 4.3.3 é discutido sobre como selecionar os k *shapelets*, que compõem a matriz da transformada, bem como sobre como definir a quantidade k.

4.3.3 O Estado da Arte

Nesta seção será detalhado o modelo estado da arte para classificação de séries temporais pelo uso da transformada *shapelet*, que é o modelo proposto por Hills *et al.* (2014). A partir deste modelo serão notados alguns problemas e limitações, que serão explorados na Seção 4.3.4.

Para o modelo de Hills *et al.* (2014) podemos começar por determinar a quantidade k de *shapelets* que serão usados na transformada. No trabalho de Lines *et al.* (2012) foram avaliadas duas heurísticas para se estimar k: por utilizar $k = \frac{m}{2}$, no qual m é o tamanho da série temporal; e por determinar k por um processo de *5-folds* no qual para cada iteração de um *fold* são induzidos m modelos (um de 1 *shapelet*, um segundo de 2 *shapelets*, e assim por diante até um m-ésimo de m *shapelets*), e ao final k é escolhido por selecionar o modelo que na média obteve o melhor resultado. A proposta de *5-fold* é custosa e na avaliação de Lines *et al.* (2012), que utilizou 26 conjuntos de dados e 7 classificadores em conjunto com a transformada, no melhor caso ela

trouxe um acréscimo médio na acurácia de 1,36% para um dos classificadores, logo tanto Lines *et al.* (2012) quanto Hills *et al.* (2014) utilizaram a proposta de $k = \frac{m}{2}$.

Para obter os k *shapelets* do total de p é utilizado um processo que tenta mitigar *shapelets* redundantes, pois *shapelets* que são similares dificilmente adicionam uma nova informação que auxilia o classificador. Por exemplo, na Figura 13 se mostra dois *shapelets* que foram extraídos da mesma região de uma mesma série temporal e por isso são similares.

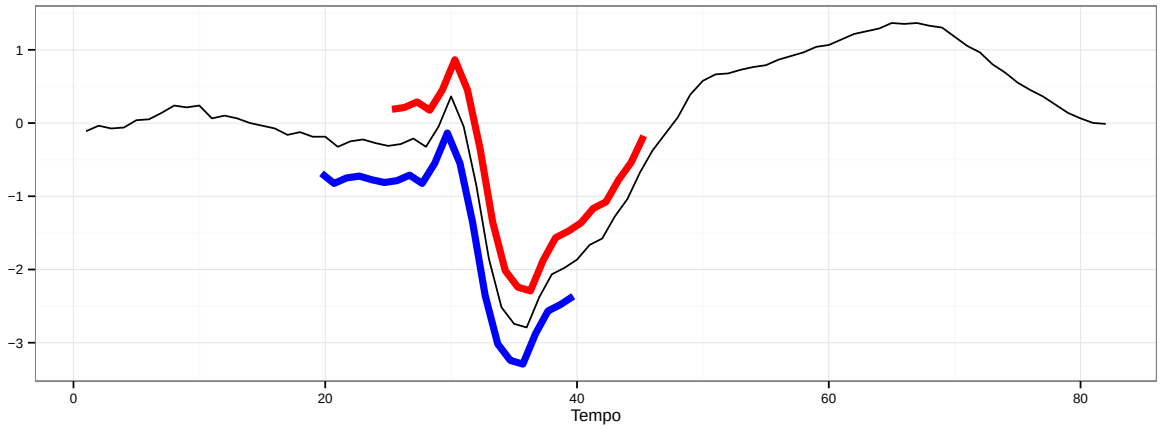

Figura 13 – Ilustração de dois *shapelets* similares extraídos de uma mesma série temporal do conjunto de dados TwoLeadECG.

Pela definição utilizada em (HILLS *et al.*, 2014), dois *shapelets* são similares se eles foram extraídos da mesma série temporal e apresentam intersecção de uma ou mais amostras. Sendo assim, para elaborar o conjunto de k *shapelets*, primeiro os ordena de forma decrescente de acordo com a medida de qualidade, então ao percorrer essa ordenação um *shapelet* é adicionado ao conjunto somente se ele não é similar a nenhum dos *shapelets* já selecionados. Este processo termina quando k *shapelets* são selecionados, e é formalizado pelo Algoritmo 4.

Algoritmo 4: Seleciona k de p *shapelets*.

Entrada: Conjunto Sh de p shapelets e a quantidade k.
Saída: Conjunto $Sh.selecionado$ de k *shapelets*.

```
1 Sh.selecionado = ∅;
2 Sh = ordenaPelaQualidade(Sh, decrescente);
3 i = 1;
4 enquanto i ≤ p E k > 0 faça
5     se similar(Sh_i, Sh.selecionado) = FALSO então
6         Sh.selecionado.adiciona(Sh_i);
7         k = k − 1;
8     i = i + 1;
9 retorna Sh.selecionado
```

Por fim, por questões de tempo de execução, o trabalho de Hills *et al.* (2014) não gera todos os possíveis *shapelets*. Como notado no Algoritmo 3 de extração de *shapelets* é preciso

fornecer os parametros de *min* e *max*, que determinam o intervalo de tamanho de subsequências a serem geradas. Hills *et al.* (2014) usam o Algoritmo 5 para estimar *min* e *max* de forma a reduzir o tempo de computação, e espera-se que sem sacrificar demasiadamente a acurácia final. Este algoritmo seleciona aleatoriamente 10 séries temporais do conjunto de treinamento, e destas se extraí todos os *shapelets* possíveis (na literatura a extração de todos os *shapelets* ocorre quando $min = 3$ e $max = m$), sendo que destes $k = 10$ são selecionados pelo processo descrito anteriormente. Este processo é repetido 10 vezes até se obter um conjunto de 100 *shapelets*, que são ordenados pelo seu tamanho e então *min* é definido como o 25º *shapelet* e *max* o 75º *shapelet*.

Algoritmo 5: Estimação de *min* e *max*. A função *ExtraiShapelets(...)* é o Algoritmo 3, enquanto que a função *SelecionaShapelets(...)* é o Algoritmo 4.

Entrada: Conjunto de dados CD.
Saída: Os parâmetros *min* e *max* que definem o intervalo de tamanho das subsequências a serem geradas.

```
1  Sh = ∅;
2  para i = 1 até 10 faça
3      CD' = amostra(CD, n = 10);
4      tmp.Sh = ExtraiShapelets(CD', min = 3, max = m);
5      tmp.Sh = SelecionaShapelets(tmp.Sh, k = 10);
6      Sh.adiciona(tmp.Sh);
7  Sh = ordenaPorTamanho(Sh);
8  min = tamanho(Sh[25]);
9  max = tamanho(Sh[75]);
10 retorna min, max
```

Resumindo todo o processo:

1. Estimar *min* e *max* pelo Algoritmo 5.
2. Extrair e computar a medida de qualidade de todos os *shapelets* de tamanho entre *min* e *max* pelo Algoritmo 3.
3. Selecionar os k melhores *shapelets* pelo Algoritmo 4.
4. Computar a transformada *shapelet*.
5. Induzir o classificador sob a transformada.

Lembrando que Hills *et al.* (2014) recomendam o uso da medida de qualidade *f-statistic* pelos seus experimentos na árvore de decisão *shapelets*, porém todos os seus experimentos realizados na transformada foram com o ganho de informação. Em nossos experimentos fazemos comparativos com o modelo estado da arte em que a medida de qualidade é tanto o ganho de informação quanto a *f-statistic*.

4.3.4 Limitações & Trabalhos Relacionados

A primeira limitação que falaremos sobre também foi notada por Hills *et al.* (2014) e consiste na detecção de *shapelets* redundantes. Da forma como a similaridade foi definida a redundância só ocorre em *shapelets* extraídos da mesma série temporal, mas se um *shapelet* descreve bem uma classe, é por que o padrão que o *shapelet* representa aparece nas séries daquela classe, ou seja, existem inúmeros *shapelets* similares mas de séries temporais diferentes e por isso passam pelo crivo de similaridade. Hills *et al.* (2014) enfrentaram esse problema pela perspectiva de interpretabilidade, ou seja, ao se analisar o conjunto de *shapelets* para entender o processo de classificação não é desejável ter inúmeros *shapelets* redundantes. A solução por eles proposta consiste em aplicar o algoritmo de agrupamento hierárquico para reduzir de k para k' a quantidade de *shapelets*, no qual cada grupo é representado pelo *shapelet* de melhor qualidade. Em seus experimentos, a acurácia ao se usar k *shapelets* é um pouco melhor do que ao se reduzir por agrupamento para k', porém a acurácia obtida por agrupar os k em k' *shapelets* é melhor do que ao se utilizar diretamente k' *shapelets*. De qualquer forma, existe um ganho subjetivo ao se analisar uma quantidade menor de *shapelets*.

O trabalho de Yuan, Wang e Han (2014) é interessante pois ataca tanto o problema de *shapelets* redundantes quanto o problema de estimar k. O seu trabalho exige que a medida de qualidade possua um ponto de corte τ atrelado ao *shapelet*, sendo assim dentre as nossas medidas avaliadas só pode ser usado pelo de ganho de informação. Para Yuan, Wang e Han (2014) um *shapelet* descrito por uma subsequência S_i, de classe c_i, com medida de qualidade q_i e com um ponto de corte τ_i é similar a outro *shapelet* $\langle S_j, q_j, \tau_j, c_j \rangle$, no qual $q_i \geq q_j$ e $c_i = c_j$, se $dist(S_i, S_j) \leq \tau_i$. Com esta definição de similaridade se percorre os *shapelets*, após a sua ordenação por qualidade, eliminando aqueles que são similares a outros de maior qualidade. Em um segundo passo, que elimina a necessidade de estimar k, se aplica um algoritmo de cobertura, no qual um *shapelet* $\langle S_i, q_i, \tau_i, c_i \rangle$ cobre uma série temporal T_i se $dist(S_i, T_j) \leq \tau_i$. O processo de cobertura começa por inicializar em zero a contagem de cobertura para todas as séries temporais então, ao começar do *shapelet* de maior qualidade para o de menor, se incrementa o contador de cobertura para cada série temporal que o *shapelet* cobre. Quando uma série temporal é coberta por α *shapelets* ela é retirada do processo, e o processo termina quando todas as séries temporais são cobertas por α vezes. Por fim, o conjunto de *shapelets* que será usado na transformada é aquele de todos os *shapelets* que cobriram pelo menos uma série temporal. Como visto, Yuan, Wang e Han (2014) oferecem soluções bem engenhosas, porém elas exigem a existência de um ponto de corte, o qual limita a sua aplicação.

Entretanto, o trabalho de Wistuba, Grabocka e Schmidt-Thieme (2015) alega que é possível atingir a mesma acurácia de um modelo que explora todos os *shapelets* por somente amostrar aleatoriamente algumas centenas de *shapelets*, sem utilizar nenhum processo de avaliação dos *shapelets*, simplesmente todos os que são amostrados são usados na transformada. Em nosso trabalho são feitos diversos experimentos que avaliam o uso da amostragem aleatória, porém

com o uso das medidas de qualidade.

Por fim, Grabocka *et al.* (2014) apresentam uma mudança de paradigma, pois ao invés de escolher bons *shapelets* o método deles inventa e aprende bons *shapelets*. Para uma breve explicação do método vamos supor o caso de um conjunto de dados de somente duas classes ($c_i \in \{0,1\}$), no qual se deseja obter bons *shapelets* de um único tamanho l. Então seja S uma subsequência, no qual cada amostra é multiplicada por um peso $W = \{w_1, w_2, \ldots, w_l\}$, em que se deseja definir W e S de forma que $f(sigmoid(w_0 + \sum_{i=1}^{l} s_i w_i)) = c$ (no qual $f(\ldots)$ é uma função que dado um valor numérico retorna 0 ou 1). Nesta formulação S é o *shapelet* que será aprendido e W é um plano, no qual ambos serão otimizados para separar linearmente todas as séries temporais. De certa forma, este modelo se parece com um perceptron, e bem como no perceptron a inicialização dos valores é importante. Para W basta inicializar com valores pequenos centrados em 0, mas para S recomenda-se uma inicialização através dos centróides de um *k-means* aplicado em todas as subsequências possíveis. Maiores detalhes são encontrados em (GRABOCKA *et al.*, 2014), no qual os autores reportaram grande acurácia, porém os *shapelets* aprendidos não aparentam garantir grande interpretabilidade.

4.4 Considerações Finais

Neste capítulo foi apresentado o conceito de *shapelets* e formalizada a transformada *shapelet*. Em conjunto com a transformada também foi detalhado o estado da arte para classificação de séries temporais pela transformada. O estado da arte de Hills *et al.* (2014) determina a quantidade de *shapelets* e quais *shapelets* devem ser usados na transformada. No Capítulo 5 serão realizados experimentos sob a transformada e com os classificadores do Capítulo 3 que irão questionar o método de seleção de *shapelets* e o método para redução do espaço de busca do estado da arte.

CAPÍTULO

5

AVALIAÇÃO EXPERIMENTAL

5.1 Considerações Iniciais

Neste capítulo são apresentados os experimentos realizados neste trabalho. A primeira seção, 5.2, é dedicada a definir como os experimentos foram executados, incluindo uma descrição dos conjuntos de dados e do ambiente computacional utilizado. Na Seção 5.3 são exibidos os resultados de experimentos que visam avaliar como as medidas de qualidade e a amostragem aleatória afetam a acurácia dos modelos finais. Enquanto que a Seção 5.4 inclui experimentos que visam a comparação com o modelo estado da arte. Dentre os experimentos estão um de avaliar o tempo necessário para executar o modelo estado da arte, em especial quanto tempo é despendido na estimação dos parâmetros *min* e *max* e o quão capazes são esses parâmetros em reduzir o espaço de busca dos *shapelets*. Outro experimento visa comparar a acurácia da redução do espaço de busca dos *shapelets* por amostragem aleatória em relação a estimação dos parâmetros *min* e *max*. E um último experimento faz uma comparação direta entre o modelo estado da arte e o modelo proposto.

5.2 Projeto Experimental

Começamos a descrição do projeto experimental pelo ambiente computacional. Todos os experimentos foram executados em um *Intel Core i7* de 4 núcleos e de 2,3GHz, e a linguagem de programação utilizada foi a R, sendo que para execução em tempo hábil dos experimentos o código foi paralelizado com o pacote *RcppParallel*[1], que não somente permite o uso de múltiplos núcleos como também a execução de código *C++*.

Além disso, todos os nossos resultados advêm da execução em 28 conjuntos de dados retirados do repositório da Universidade da Califórnia em Riverside (UCR) (CHEN *et al.*, 2015).

[1] http://rcppcore.github.io/RcppParallel/

Uma descrição de cada um desses conjuntos de dados é dada na Tabela 4.

Tabela 4 – Descrição dos 28 conjuntos de dados utilizados nos experimentos. Todos eles estão disponíveis no repositório da Universidade de Califórnia em Riverside (UCR) (CHEN *et al.*, 2015).

Nome	Nº de classes	Nº de séries temporais de treinamento	Nº de séries temporais de teste	Nº de amostras por série temporal
TwoLeadECG	2	23	1139	82
Gun-Point	2	50	150	150
ECGFiveDays	2	23	861	136
MoteStrain	2	20	1252	84
SonyAIBORobot Surface	2	20	601	70
ToeSegmentation1	2	40	228	277
MiddlePhalanxOutlineCorrect	2	291	600	80
DistalPhalanxOutlineCorrect	2	276	600	80
Wine	2	57	54	234
Coffee	2	28	28	286
ToeSegmentation2	2	36	130	343
ShapeletSim	2	20	180	500
BeetleFly	2	20	20	512
CBF	3	30	900	128
MiddlePhalanxOutlineAgeGroup	3	154	400	80
DistalPhalanxOutlineAgeGroup	3	139	400	80
ArrowHead	3	36	175	251
DiatomSizeReduction	4	16	306	345
Face (four)	4	24	88	350
Trace	4	100	100	275
Beef	5	30	30	470
ProximalPhalanxTW	6	205	400	80
Symbols	6	25	995	398
Synthetic Control	6	300	300	60
Plane	7	105	105	144
MedicalImages	10	381	760	99
FacesUCR	14	200	2050	131
Swedish Leaf	15	500	625	128

Durante a execução dos experimentos notou-se a necessidade de alguns cuidados e modificações nos Algoritmos 4 e 5. O primeiro cuidado é o que chamamos de *bias* da ordem de computação dos *shapelets*. É um problema que ocorre principalmente na execução do Algoritmo 4 que seleciona os k melhores *shapelets* e consiste no fato de que é possível diferentes *shapelets* obterem o mesmo valor de qualidade, logo é importante levar em consideração a estabilidade do algoritmo de ordenação. Sempre que um conjunto de *shapelets* é ordenado, nós tomamos o cuidado de embaralhar eles aleatoriamente. Outro cuidado ocorre no Algoritmo 5, que faz a estimação dos parâmetros *min* e *max*. Ao se amostrar 10 séries aleatoriamente nós garantimos que existem séries de pelo menos duas classes (o conjunto é re-amostrado até contemplar esta restrição).

Ao final da Seção 4.3.3, foi dada a sequência de passos utilizada pelo estado-da-arte para classificar um conjunto de dados. Nesta seção nós redefinimos esta sequência passos para acomodar outros casos de extração de *shapelets*, por exemplo, ao se amostrar aleatoriamente os *shapelets*. Dessa maneira, os passos utilizados na avaliação deste capítulo são:

1. Extração dos *shapelets*, que pode ser por:

   - Estimar *min* e *max* pelo Algoritmo 5 e então realizar a extração dos *shapelets* pelo Algoritmo 3. Ou,

- Extrair todos os *shapelets* pelo Algoritmo 3 ($min = 3$ e $max = m$). Ou,
- Extrair todos os *shapelets* pelo Algoritmo 3 ($min = 3$ e $max = m$) e selecionar aleatoriamente uma proporção r destes (na prática estes dois processos são feitos em conjunto).

2. Do conjunto de *shapelets* obtidos selecionar k pelo Algoritmo 4.
3. Computar a transformada *shapelet* como descrito na Seção 4.3.2.
4. Induzir o classificador sob a transformada.

Neste trabalho todas as avaliações são feitas com 3 classificadores, e cada um deles requer a definição de pelo menos um parâmetro:

kNN: para este classificador $k = 1$, no qual k é a quantidade de vizinhos utilizados para definir a classe, como explicado na Seção 3.2. Além disso é utilizada a distância euclidiana da Equação 2.1. A implementação utilizada é a do pacote *class* versão 7.3-13.

Random Forest: para este classificador é preciso definir a quantidade de árvores da floresta, que definimos em 5 vezes a quantidade de séries temporais no conjunto de treinamento. Todos os outros parâmetros são deixados com seus valores padrões. A implementação utilizada é a do pacote *randomForest* versão 4.6-10.

SVM: o principal parâmetro deste classificador é o *kernel*, que foi definido em linear para fins de continuidade e comparação com os trabalhos prévios de Lines *et al.* (2012), Hills *et al.* (2014). Todos os outros parâmetros são deixados com seus valores padrões. A implementação utilizada é a do pacote *e1071* versão 1.6-7.

Por fim, decidimos adotar uma convenção diferente para a quantidade de atributos que varia de acordo com o conjunto de dados. A intuição é que para classificar corretamente C classes é preciso C atributos, então a nossa quantidade k é definida como um múltiplo de C: $k = k' \times C$. Com esta convenção nós descartamos a recomendação de $k = \frac{m}{2}$, porém como veremos em nosso primeiro experimento, nós determinamos a quantidade k' que maximiza acurácia para todas as medidas de qualidade em todos os classificadores, tornando assim a comparação a mais justa possível.

5.3 Avaliação da Acurácia

Nesta seção são apresentados dois experimentos. O primeiro deles visa avaliar como as medidas de qualidade impactam a acurácia em diferentes níveis de quantidade de atributos, no qual todos os *shapelets* são utilizados para evitar influência de fatores externos. Além disso, este

experimento permite determinar em qual quantidade de atributos a acurácia é maximizada para as medidas de qualidade nos três classificadores. Enquanto que no segundo experimento se é avaliado como a amostragem aleatória de *shapelets* em diferentes níveis de proporção afeta a acurácia dos modelos finais.

5.3.1 Avaliação das Medidas de Qualidade

Em nosso primeiro experimento desejamos avaliar como as medidas de qualidade impactam a acurácia, para isso este experimento extraí todos os *shapelets* possíveis e mede a acurácia dos classificadores em diferentes níveis de quantidade de atributos. Como benefício extra deste experimento temos o de obter a quantidade de atributos em que a acurácia atinge um plateau para cada uma das medidas de qualidade, assim abandonando de forma segura a recomendação de $k = \frac{m}{2}$. O resultado deste experimento é mostrado na Figura 14, no qual cada ponto é a acurácia média de todos os conjuntos de dados, sendo que para cada conjunto de dados foram feitas 10 execuções.

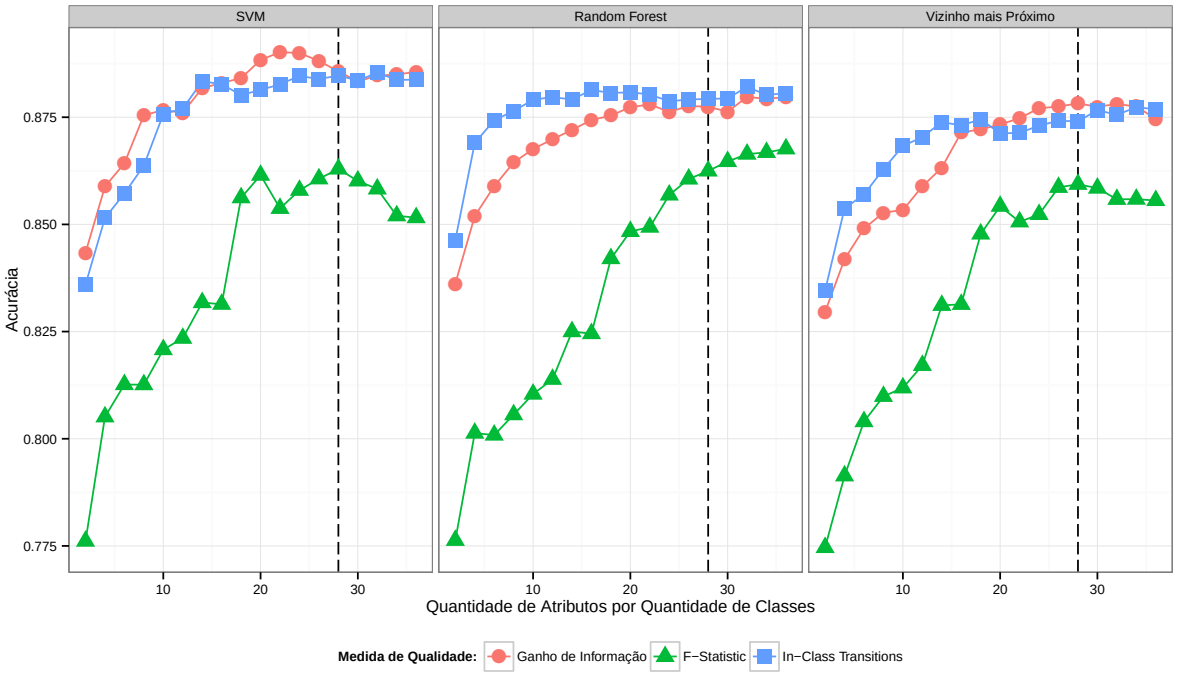

Figura 14 – Média da acurácia em todos os conjuntos de dados com variação da quantidade de atributos por quantidade de classes. Todas as medidas de qualidade atingem um plateau quando a quantidade de atributos por classe é 28 (linha tracejada).

Da Figura 14 é possível notar que não importa o classificador ou a quantidade de atributos a medida de qualidade *f-statistic* possui sempre a pior acurácia. Além disso, entre 16 e 18 atributos por quantidade de classes é possível notar um salto na acurácia da *f-statistic*. Ao inspecionar este salto notamos que para o conjunto de dados Symbols ocorre uma mudança brusca na acurácia de cerca de 30% para próximos de 80% nos 3 classificadores (ver Tabela 5). Indo além, ao se buscar por quais *shapelets* que compõem os atributos até a marca de 16 atributos por quantidade de

classes se nota que existe somente um tipo de *shapelet*: o de uma linha reta, que classifica muito bem uma das 6 classes. Esta é a manifestação do problema de *shapelets* redundantes oriundos de diferentes séries temporais. O que acontece é que se um *shapelet* possui alta qualidade, então é esperado que os outros *shapelets* similares a ele também tenham e assim ocupem as primeiras posições, evitando que outros *shapelets* mais diversos possam dar uma maior contribuição ao processo de classificação. A menos que se retirem esses *shapelets* redundantes uma solução é aumentar a quantidade de atributos para incluir outros que sejam diferentes. Curiosamente, esperávamos que esse problema fosse se manifestar para as outras duas medidas, pois ambas possuem níveis de qualidade bem definidos proporcionando maior possibilidade de empate na métrica de qualidade, enquanto que a *f-statistic* é contínua nos números reais.

Tabela 5 – Acurácia para o conjunto de dados Symbols quando a medida de qualidade *f-statistic* é utilizada. Note como na Figura 14 ocorre um salto na acurácia da *f-statistic* entre 16 e 18 atributos por quantidade de classes, que se deve a uma mudança brusca da acurácia para o conjunto de dados Symbols, como mostra essa tabela.

	Quantidade de Atributos por Quantidade de Classes									
Classificador	6	8	10	12	14	16	18	20	22	24
SVM	0,29	0,29	0,29	0,29	0,29	0,29	0,79	0,79	0,80	0,84
Random Forest	0,29	0,29	0,29	0,29	0,29	0,30	0,72	0,82	0,82	0,86
1-NN	0,29	0,29	0,29	0,29	0,29	0,32	0,79	0,81	0,82	0,87

No entanto, o mais importante do gráfico da Figura 14 é que a acurácia para todas as medidas de qualidade e em todos os classificadores atinge um plateau quando a quantidade de atributos por quantidade de classes é 28, sendo assim, para todos os outros experimentos será usado esse valor.

Apesar do gráfico da Figura 14 ser um bom indicativo de qual medida de qualidade possui a melhor performance em relação a acurácia, o uso da média da acurácia em diversos conjuntos de dados pode ocultar sua baixa acurácia em vários outros conjuntos de dados. Por isso também foi realizada uma análise do *average rank*, que em linhas gerais conta quantas vezes cada medida obteve a melhor, segunda melhor ou pior acurácia em cada um dos conjuntos de dados. O gráfico da média do *average rank* em todos os conjuntos de dados ao longo de vários níveis de atributos é dado pela Figura 15.

Do gráfico da Figura 15 confirma-se claramente que a medida de qualidade *f-statistic* possui a pior performance, o que é algo interessante dado os experimentos de Hills *et al.* (2014) na árvore de decisão *shapelet*, que foram satisfatórios o suficiente a ponto dos autores recomendarem o uso da *f-statistic* como medida de qualidade padrão. Além do mais, a nossa medida proposta domina os resultados quando poucos atributos são utilizados, e ao se incrementar a quantidade de atributos então a medida de ganho de informação se torna competitiva com a *in-class transitions*.

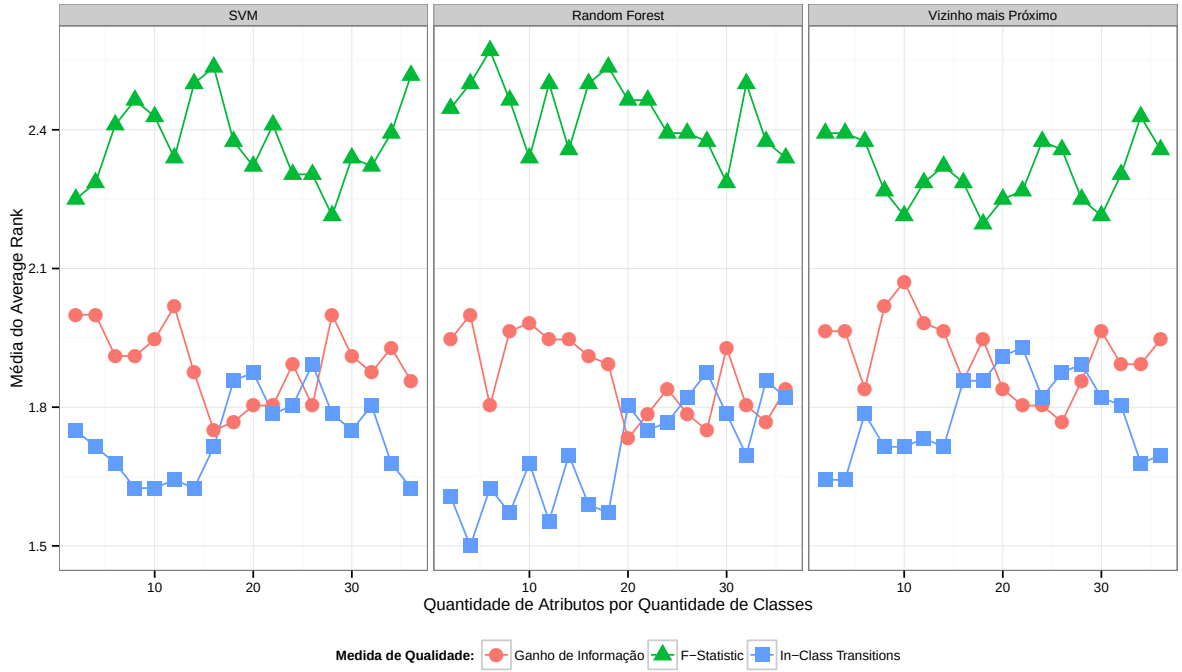

Figura 15 – Média do *average rank* em todos os conjuntos de dados com variação da quantidade de atributos por quantidade de classes.

5.3.2 Avaliação da Amostragem Aleatória

O segundo experimento foi executado com a quantidade de atributos por quantidade de classes fixada em 28, e os resultados também são provenientes de 10 execuções. Neste experimento é avaliado como a amostragem aleatória de *shapelets* afeta a acurácia, pois caso a acurácia não sofra grande degradação, então isto pode se tornar uma estratégia viável para minimizar o tempo de execução. A estratégia de amostragem aleatória dos *shapelets* foi avaliada nos níveis de 0,05%, 0,1%, 0,5%, 1%, 2,5%, 5%, 10%, 25% e 100% (sem amostragem, para fins de comparação). Os resultados são exibidos nos gráficos das Figuras 16 e 17.

O gráfico da Figura 17 é o mesmo da Figura 16, mas apenas muda a faixa de valores da acurácia para maior de 50%. A importância da Figura 16 está em mostrar que para determinadas proporções de amostragem o conjunto de dados ToeSegmenation2 possui baixa acurácia nas medidas de qualidade ganho de informação e *in-class transitions*. De qualquer forma, como estamos interessados no comportamento geral da acurácia ao realizar amostragem aleatória, o gráfico da Figura 17 provém maiores detalhes. Deste gráfico é possível notar pela mediana que amostrar costuma trazer uma influência positiva na acurácia, já que raramente não amostrar (amostragem em 100%) possui a melhor acurácia. Nossa hipótese para esse fenômeno é que a amostragem está proporcionado maior diversidade nos atributos, pois a amostragem retira indiretamente *shapelets* redundantes, evitando assim o problema ocorrido com a *f-statistic* no conjunto de dados Symbols e auxiliando os classificadores. Mais do que isso, mesmo ao se amostrar a um nível baixo de 0,05% não é possível detectar uma degradação significante na

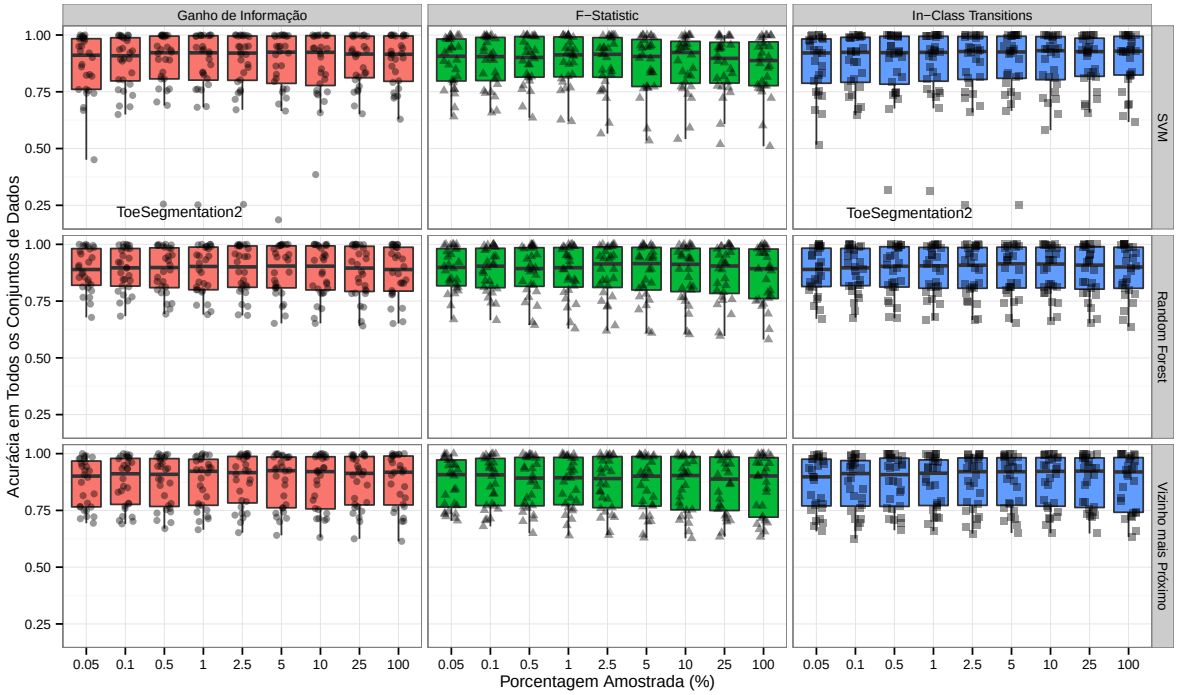

Figura 16 – Conjunto de Dados ToeSegmentation2 possui baixa acurácia para as medidas de qualidade ganho de informação e *in-class transitions* entre os níveis de 0,5% e 10% de amostragem. A Figura 17 mostra o mesmo gráfico mas com enfoque na região de acurácia entre 50% e 100%, proporcionado melhor compreensão do impacto da amostragem aleatória na acurácia dos classificadores.

acurácia, sendo assim amostragem é uma excelente técnica para redução do tempo de execução sem sacrificar a acurácia. Por fim, notamos que o melhor balanço entre tempo de execução e acurácia costuma ocorrer quando 5% dos *shapelets* são amostrados.

5.4 Avaliação do Estado da Arte

Uma vez feito experimentos que comprovaram a eficiência da nossa medida de qualidade e que determinaram a quantidade de atributos e o nível de amostragem em que as medidas de qualidade possuem sua melhor performance, estamos aptos a fazer experimentos que comparem com o estado da arte.

Tanto a Seção 5.4.1 como a Seção 5.4.2 tem como objetivo principal descartar o uso do Algoritmo 5 de estimação de *min* e *max* em favor do uso da amostragem aleatória. Por fim, na Seção 5.4.3, é feita uma comparação direta entre o modelo estado da arte (uso do Algoritmo 5 com a medida de qualidade de ganho de informação ou *f-statistic*) e o modelo proposto por nós (uso de amostragem aleatória de 5% com a medida de qualidade *in-class transitions*).

5.4.1 Avaliação do Tempo de Execução

Como primeiro experimento para avaliar o uso do Algoritmo 5 nós desejamos avaliar a hipótese de que esse algoritmo tem um tempo de execução alto demais para justificar o seu uso

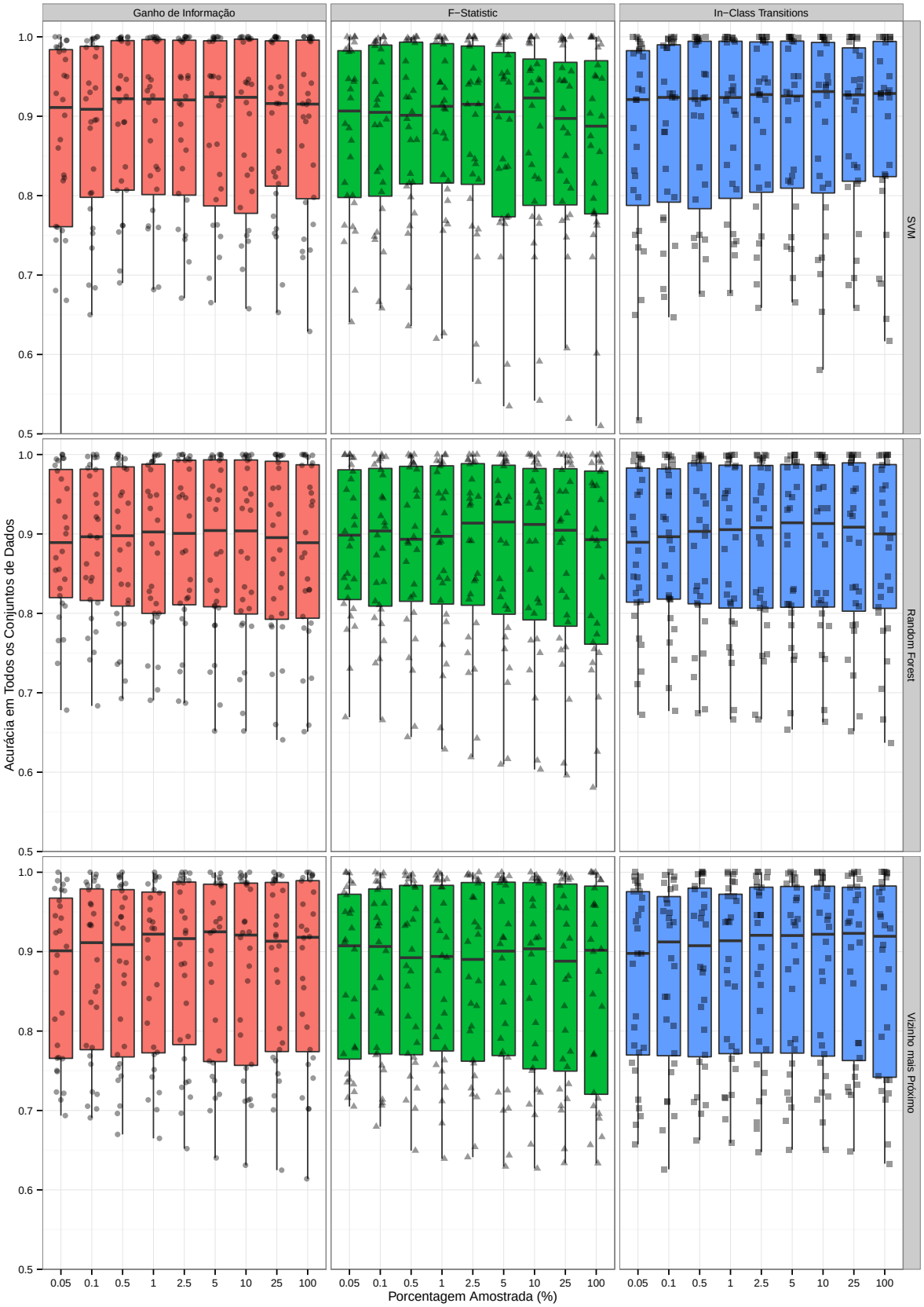

Figura 17 – Efeito da amostragem aleatória de *shapelets* na acurácia dos classificadores. Do gráfico é possível notar, ao se analisar pela mediana, que a melhor performance ocorre com amostragem de 5% dos *shapelets*. Curiosamente, não amostrar (100%) não costuma obter os melhores resultados.

prático. A nossa hipótese é baseada na observação de que ele extraí todos os *shapelets* possíveis de 10 séries temporais por 10 vezes, ou seja, um limite superior para sua execução é a de extrair todos os *shapelets* de um conjunto de dados com 100 séries temporais, e depois disso ainda é preciso extrair os *shapelets* de todas as séries temporais do conjunto de dados nos limites por ele apontado. Nós acreditamos que para conjuntos de dados de menos de 100 séries temporais o seu custo é demasiadamente caro.

Para testar essa hipótese nós tomamos o tempo para extrair todos os *shapelets* e o tempo para executar o algoritmo de estimação para cada um dos conjuntos de dados. Notamos aqui que esse processo foi executado apenas uma única vez para cada conjunto de dados, porém, argumentamos que como o tempo de execução é grande, qualquer peculiaridade durante a execução que possa ter afetado o tempo de execução não teria grande efeito.

A Tabela 6 contêm os dados deste experimento, que são: nome do conjunto de dados, quantidade de séries temporais de treinamento, tamanho da série temporal, tempo para extrair todos os *shapelets* (em segundos), tempo para executar o Algoritmo 5 (em segundos) e, por fim, a razão entre o tempo para executar o Algoritmo 5 pelo tempo para extrair todos os *shapelets*. Os dados mostram que executar o Algoritmo 5 tem um custo equiparável a extrair todos os *shapelets* de um conjunto de dados de cerca de 30 séries temporais (note a razão de quase 1 para os conjuntos de dados CBF e Beef, no qual ambos possuem 30 séries temporais mas de diferentes tamanhos).

Porém, o efeito da execução do Algoritmo 5 não para nesse pré-processamento, pois ele indica o tamanho de *min* e *max* que determina quantos *shapelets* serão extraídos posteriormente. Logo, se esse algoritmo for capaz de apresentar um intervalo pequeno e com boa acurácia, a sua execução pode continuar a ser justificada. No entanto, a Tabela 7, que mostra uma sumarização da proporção de *shapelets* que esse algoritmo indica a extração, mostra que ele tem um baixo poder de poda. No melhor caso, que ocorre para a medida de qualidade *f-statistic*, ele indica a extração de 1,5%, mas logo no primeiro quartil ele já indica a extração de mais de 10%, o que já é o dobro da nossa indicação de 5% para amostragem aleatória (que na verdade poderia até mesmo ser 0,05%, já que não se notou nenhuma grande degradação na acurácia). Porém, para descartar o uso é preciso avaliar se o Algoritmo 5 não possui uma influência positiva na acurácia.

5.4.2 Avaliação em Relação a Amostragem Aleatória

Para testar se o Algoritmo 5 possui algum efeito positivo na acurácia em relação a amostragem aleatória, é feito um experimento no qual cada medida de qualidade compete contra si mesma, com a única diferença de que em um dos casos os *shapelets* são provenientes de amostragem aleatória ao nível de 5% e no outro através do Algoritmo 5. Bem como nos experimentos anteriores, a quantidade de atributos por quantidade de classes é fixada em 28, e por questões de tempo de execução do Algoritmo 5 os resultados exibidos nos gráficos da Figura 18 são provenientes da média de 5 execuções.

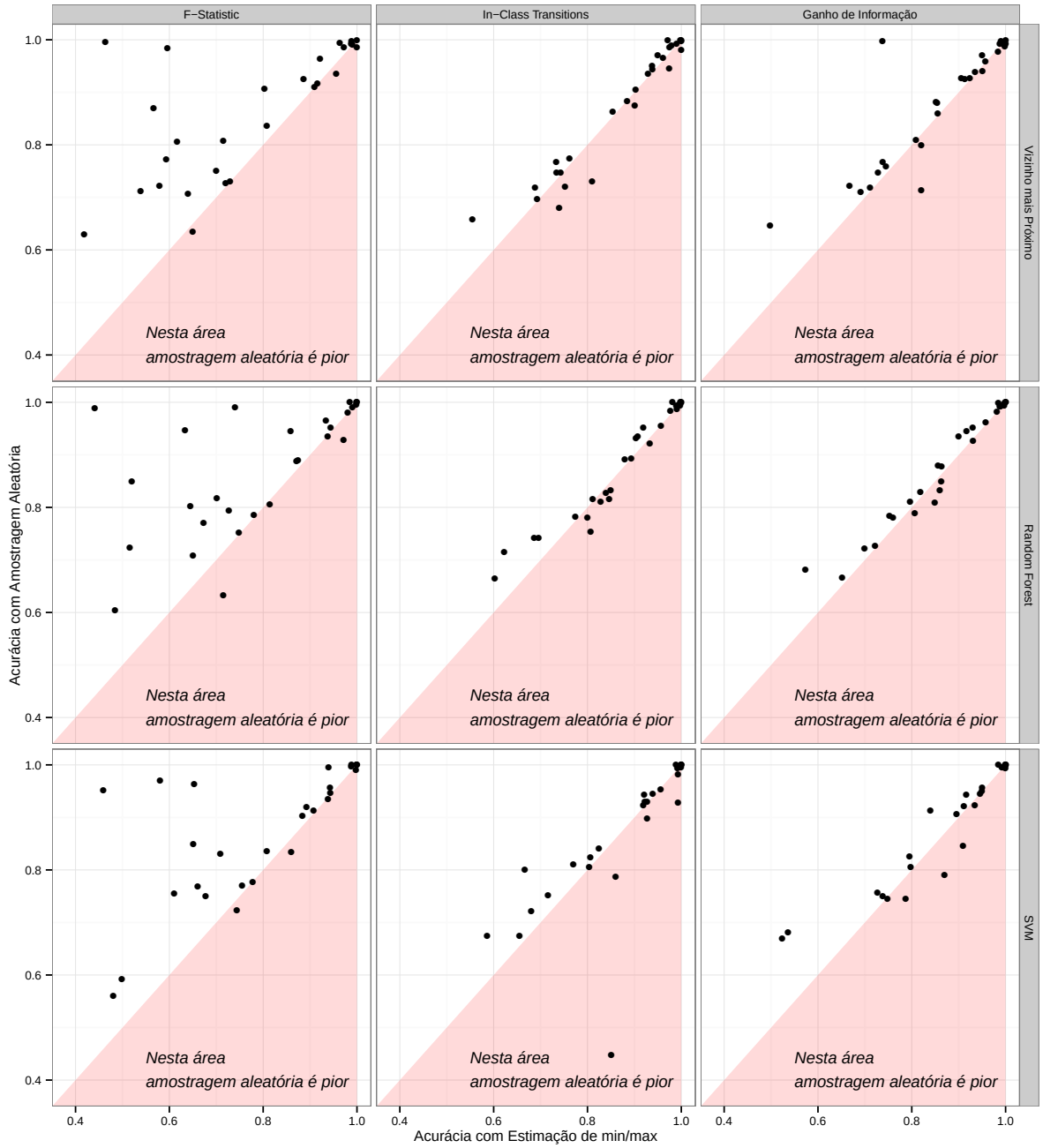

Figura 18 – Avaliação de como as técnicas de redução do espaço de busca dos *shapelets* por amostragem aleatória ou execução do Algoritmo 5 de estimação de *min* e *max* afetam a acurácia. Neste gráfico cada medida de qualidade em cada um dos classificadores compete contra si mesma, no qual a única diferença é a técnica de redução de espaço. No geral a amostragem aleatória costuma ter um desempenho um pouco melhor. A exceção é para a medida de qualidade *f-statistic*, que se beneficia fortmente da amostragem aleatória.

Tabela 6 – Avaliação do tempo de execução entre extrair todos os *shapelets* e estimar os parâmetros *min* e *max*. As três primeiras colunas detalham os conjuntos de dados, em especial a quantidade de séries de treinamento e o tamanho delas, que juntos determinam a quantidade de *shapelets*, enquanto que as 2 seguintes colunas detalham o tempo de execução de extrair todos os *shapelets* e estimar os parâmetros de *min* e *max* e, por fim, a última coluna é simplesmente uma razão entre as duas colunas anteriores. A partir destes dados observa-se que a execução do algoritmo de estimação de *min* e *max* tem custo similar a de se extrair todos os *shapelets* de um conjunto de dados com cerca de 30 séries temporais.

Nome	Nº de séries de treinamento	Tamanho da série	Tempo para extrair todos os *shapelets* (s)	Tempo para Estimar *min* e *max* (s)	Razão
DiatomSizeReduction	16	345	497,57	1892,37	3,80
SonyAIBORobot Surface	20	70	1,64	4,54	2,76
MoteStrain	20	84	3,27	8,88	2,72
ShapeletSim	20	500	3419,73	8324,25	2,43
BeetleFly	20	512	3536,89	8939,84	2,53
TwoLeadECG	23	82	3,98	7,97	2,00
ECGFiveDays	23	136	28,58	53,51	1,87
Face (four)	24	350	1178,77	2071,51	1,76
Symbols	25	398	2105,82	3354,47	1,59
Coffee	28	286	718,83	922,05	1,28
CBF	30	128	37,95	43,60	1,15
Beef	30	470	6371,80	6374,94	1,00
ArrowHead	36	251	709,57	570,14	0,80
ToeSegmentation2	36	343	2432,21	1852,02	0,76
ToeSegmentation1	40	277	1302,62	840,64	0,65
Gun-Point	50	150	197,51	76,29	0,39
Wine	57	234	1347,20	432,91	0,32
Trace	100	275	7807,10	812,28	0,10
Plane	105	144	708,65	69,33	0,10
DistalPhalanxOutlineAgeGroup	139	80	133,48	7,47	0,06
MiddlePhalanxOutlineAgeGroup	154	80	163,00	7,30	0,04
FacesUCR	200	131	1845,21	47,21	0,03
ProximalPhalanxTW	205	80	285,31	8,27	0,03
DistalPhalanxOutlineCorrect	276	80	508,74	7,32	0,01
MiddlePhalanxOutlineCorrect	291	80	571,94	7,21	0,01
Synthetic Control	300	60	221,99	3,47	0,02
MedicalImages	381	99	2199,22	16,67	0,01
Swedish Leaf	500	128	10353,86	41,82	0,00

Tabela 7 – Sumarização das porcentagens de *shapelets* a serem extraídas de acordo com a estimava de *min* e *max* para as diversas medidas de qualidade nos conjuntos de dados da Tabela 4.

Medida de Qualidade	Mínimo	1º Quartil	Mediana	Média	3º Quartil	Máximo
Ganho de Informação	25,7	31,51	35,31	37,24	43,94	56,09
F-Statistic	1,558	10,56	16,08	18,28	27,81	40,26
In-Class Transitions	20,11	31,3	36,52	36,28	41,76	49,08

Os resultados mostram que para as medidas de qualidade ganho de informação e *in-class transitions* costuma ocorrer um ganho de alguns pontos percentuais de acurácia ao se usar amostragem aleatória, porém, para a medida *f-statistic* o ganho é bem mais acentuado. Com isso concluímos que o uso de amostragem aleatória tende a melhorar o tempo de execução e também a acurácia final.

5.4.3 Comparação com o Nosso Modelo Proposto

Por fim fazemos uma comparação direta entre o nosso modelo proposto (amostragem aleatória de 5% com uso da medida de qualidade *in-class transitions*) contra o modelo estado da arte (uso do Algoritmo 5 com as outras duas medidas de qualidade). Para ambos os modelos é usado 28 atributos por quantidade de classes, e os resultados aqui exibidos são fruto de 5 execuções.

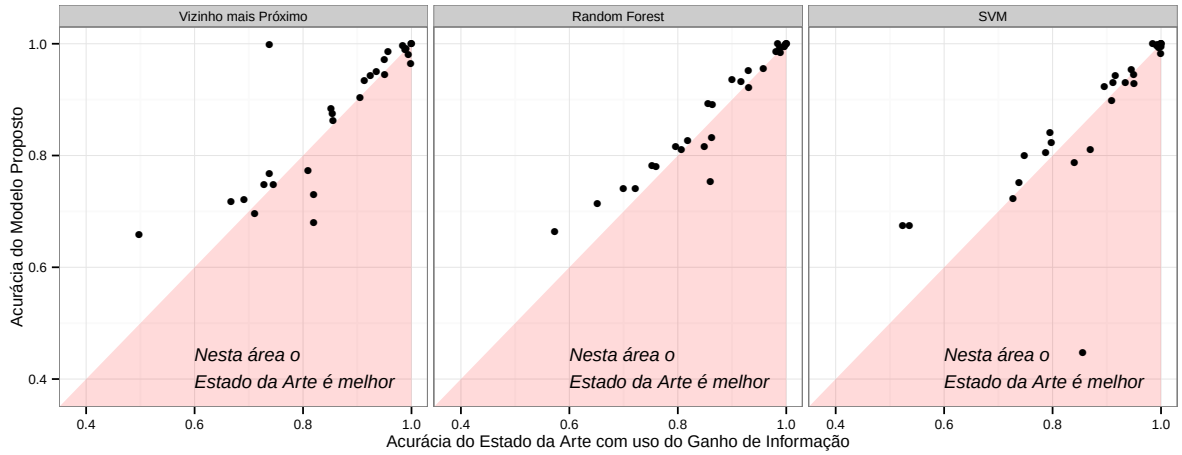

Figura 19 – Comparação entre o modelo Estado da Arte (usando como medida de qualidade o ganho de informação) e o nosso modelo proposto. Na maioria dos casos o nosso modelo proposto possui maior acurácia, porém sem significância estatística (ver Tabela 8).

A Figura 19 mostra a comparação descrita quando o modelo Estado da Arte usa como medida de qualidade o ganho de informação. A Tabela 8 resume os resultados, que mostram que o modelo proposto tem uma maior acurácia na maioria dos casos, porém a diferença na acurácia não é significativa o suficiente em um *t-test* pareado de 95% de confiança (para todos os classificadores).

Tabela 8 – Sumarização dos resultados da Figura 19. Esta tabela mostra quantas vezes o modelo proposto obteve uma acurácia melhor, pior ou empatou em relação ao modelo Estado da Arte que usa como medida de qualidade o ganho de informação. Apesar dos resultados favoráveis ao nosso modelo proposto, a diferença de acurácia não é significativa em um *t-test* pareado de 95% de confiança.

Nearest Neighbor			*Random Forest*			SVM		
Vitórias	Derrotas	Empates	Vitórias	Derrotas	Empates	Vitórias	Derrotas	Empates
18	8	2	18	7	3	15	11	2
Intervalo de 95% de Confiança								
-0,0122	-	0,0410	-0,0035	-	0,0230	-0,0368	-	0,0338

Já a Figura 20 mostra a comparação descrita quando o modelo Estado da Arte usa como medida de qualidade a *f-statistic*. A Tabela 9 sumariza os resultados, que mostram que o modelo proposto tem uma maior acurácia na maioria dos casos, e desta vez a diferença de acurácia é significativa em um *t-test* pareado de 95% de confiança (para todos os classificadores).

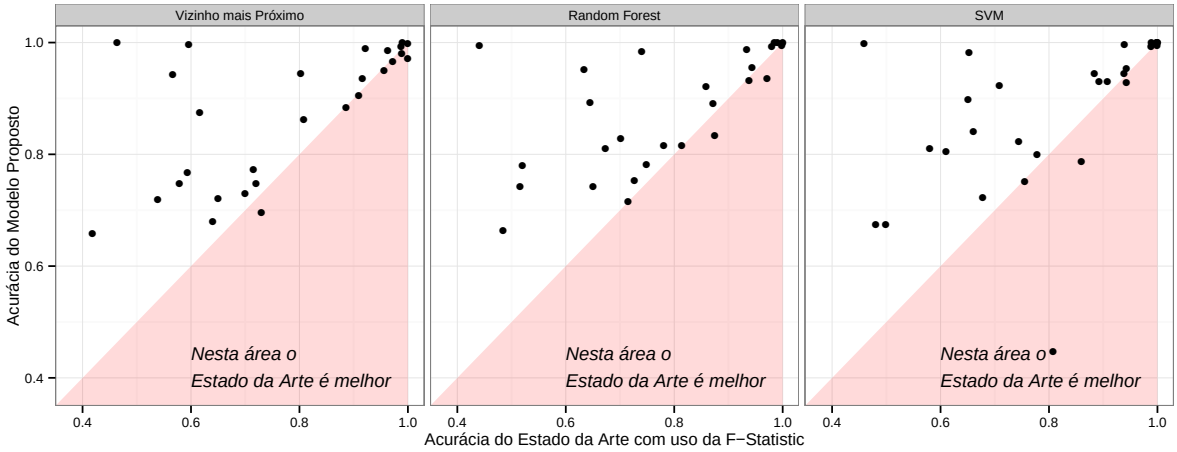

Figura 20 – Comparação entre o modelo Estado da Arte (usando como medida de qualidade a *f-statistic*) e o nosso modelo proposto. Na maioria dos casos o nosso modelo proposto possui maior acurácia, e desta vez com significância estatística (ver Tabela 9).

Tabela 9 – Sumarização dos resultados da Figura 20. Esta tabela mostra quantas vezes o modelo proposto obteve uma acurácia melhor, pior ou empatou em relação ao modelo Estado da Arte que usa como medida de qualidade a *f-statistic*. Os resultados favoráveis ao nosso modelo são confirmados por um *t-test* pareado de 95% de confiança.

Nearest Neighbor			*Random Forest*			SVM		
Vitórias	Derrotas	Empates	Vitórias	Derrotas	Empates	Vitórias	Derrotas	Empates
20	8	0	21	5	2	21	5	2
Intervalo de 95% de Confiança								
0,0434	-	0,1561	0,0394	-	0,1449	0,0175	-	0,1403

5.5 Considerações Finais

Neste capítulo fizemos uma série de experimentos para validar as nossas hipóteses. Começamos por avaliar as 3 medidas de qualidades em um ambiente que utiliza todos os *shapelets* possívies, assim excluindo efeitos de amostragem. No gráfico da Figura 14 que mostra a acurácia e no gráfico da Figura 15 que mostra o *average rank* ao se variar a quantidade de atributos utilizada notamos que a *f-statistic* possui a pior performance, e a *in-class transitions* domina quando poucos atributos são utilizados, mas o ganho de informação consegue equalizar a performance da *in-class transitions* quando mais atributos são utilizados.

No experimento seguinte avaliamos como a amostragem aleatória afeta a acurácia dos classificadores nas 3 medidas de qualidade. Para nossa surpresa, notamos que não amostrar não possui melhor a performance, sendo que a melhor performance ocorre com cerca de 5% de amostragem, mas mesmo ao se amostrar 0,05% não foi possível notar uma grande degradação da acurácia. Nossa suspeita para que a amostragem tenha uma influência positiva na acurácia é que ela força maior diversidade no conjunto de *shapelets*. De fato, acreditamos que a amostragem aleatória deva ser o *baseline* para comparação com técnicas que buscam forçar uma maior diversidade de atributos ou reduzir a redundância de *shapelets*.

Em outro experimento realizado nós avaliamos como o Algoritmo 5, que busca reduzir o espaço de busca de *shapelets* e por isso diminuir o tempo de execução, se compara com a simples técnica de amostragem aleatória. Não somente expomos o seu custo de execução como também notamos que ele não costuma trazer grande redução no espaço de busca. No melhor caso o Algoritmo 5 reduziu a cerca de 2%, mas no pior caso chega a quase 60%, o que é consideravelmente maior do que a nossa recomendação de 5% de amostragem aleatória. Por fim, na comparação da Figura 18, que avalia como a amostragem aleatória influência a acurácia em relação ao Algoritmo 5, notamos que no caso geral a amostragem aleatória não oferece uma piora na performance. Sendo assim, fazemos uma segunda recomendação: amostragem aleatória também deve ser usada como o *baseline* para redução do espaço de busca, pois além de ser, em termos práticos, $O(1)$, ela não apresentou degradação na acurácia em relação ao Algoritmo 5.

Em nosso último conjunto de experimentos avaliamos o nosso modelo proposto contra o modelo estado da arte. O modelo proposto é definido pelo uso de amostragem aleatória em 5% com uso da medida de qualidade *in-class transitions*, enquanto que o modelo estado da arte é definido pelo uso do Algoritmo 5 em conjunto com as outras duas medidas. Não importa com qual medida ou com qual classificador, o nosso modelo proposto possui a maior acurácia na maioria dos conjuntos de dados, e no caso da *f-statistic* o nosso modelo possui diferença significativa na acurácia (em 95% de confiança).

CAPÍTULO

6

CONCLUSÃO

O objetivo deste trabalho é o de estender o trabalho desenvolvido por Hills *et al.* (2014) sobre a transformada *shapelet*. No trabalho original, o método é direcionado por uma busca que tenta balancear tempo de execução por acurácia ao limitar a procura por bons *shapelets* àqueles que tenham um determinado tamanho. Apesar de concordamos de que é preciso um método para acelerar o processo de computação da transformada, nós acreditamos que é perigoso fazer quaisquer asserções na ausência de experimentos que utilizam todos os *shapelets* possíveis e assim evidenciariam qualquer impacto da redução do espaço de busca.

Para compor a transformada *shapelet* é preciso escolher bons *shapelets*, pois utilizar muitos ou poucos *shapelets* pode causar *overfitting* ou *underfitting*, logo é uma tarefa de importância. A escolha se faz por ranquear os *shapelets* de acordo com uma medida de qualidade, que tradicionalmente é a de ganho de informação, porém, experimentos recentes mostraram que a medida *f-statistic* pode reduzir o tempo de execução sem degradação na acurácia. No entanto, estes experimentos foram conduzidos na árvore de decisão *shapelets*, logo é natural questionar se os resultados na árvore de decisão se mantêm verdadeiros na transformada. Em nossos experimentos avaliamos essas duas medidas de qualidade e a nossa proposta, denominada *in-class transitions*, em um ambiente no qual todos os *shapelets* foram extraídos. Contrariando a sugestão de uso da *f-statistic* por Hills *et al.* (2014), nossos resultados mostram que a *f-statistic* possui a pior performance em termos de acurácia, enquanto que a *in-class transitions* possui a melhor performance quando poucos *shapelets* são utilizados, e quando muitos *shapelets* são utilizados tanto a *in-class transitions* quanto a de ganho de informação possuem performances equiparáveis. Tanto a medida de qualidade *in-class transitions* quanto a de ganho de informação necessitam de uma ordenação dos elementos para serem computadas, porém, a *in-class transitions* não somente é mais fácil de programar como também não utiliza funções custosas como de computação de logaritmo, logo, seria interessante a execução de experimentos que avaliam o tempo de execução dessas medidas para saber se a *in-class transitions* é tão lenta quanto a de ganho de informação, como Hills *et al.* (2014) mostrou em relação a *f-statistic*. De qualquer forma, se acurácia é o

fator mais importante, recomendamos o uso da *in-class transitions*.

Como dito anteriormente, existe a necessidade de acelerar o processo de computação da transformada, pois extrair e avaliar todos os *shapelets* nem sempre é viável. Tanto Hills *et al.* (2014) quanto nós abordamos este processo por reduzir o espaço de busca dos *shapelets*. Hills *et al.* (2014) propuseram um algoritmo de estimação dos parâmetros *min* e *max* que restrigem a busca por bons *shapelets* àqueles que tenham o tamanho entre esses parâmetros, enquanto que nós propusemos o uso da amostragem aleatória. Um de nossos experimentos teve como foco expor que a estimação dos parâmetros *min* e *max* tem um tempo que não é negligenciável. Além disso, nós mostramos que no melhor dos casos, para um conjunto de dados, a estimação dos parâmetros *min* e *max* reduziu o espaço de busca a 2% do original, mas no pior caso a 60%, ou seja, não há qualquer garantia de sua eficiência. Em contrapartida, o nosso experimento de uma simples amostragem aleatória mostrou que com somente 5% é possível obter bons resultados, e mesmo uma amostragem de 0,05% não mostrou grande degradação da acurácia. Ademais, em um ambiente em que foi controlado todos os fatores e variado o modo de redução do espaço de busca, o método mais custoso de estimação de parâmetros não apresentou melhor acurácia. Assim sugerimos que a amostragem aleatória se torne o *baseline* para métodos de redução do espaço de busca.

Sobre a amostragem aleatória ocorreu um fenômeno curioso. A acurácia de modelos com amostragem aleatória se mostrou melhor do que modelos que utilizaram todos os *shapelets*. Nossa hipótese é de que a amostragem força uma maior diversidade nos *shapelets*, auxiliando assim os classificadores a obter uma maior acurácia. Este problema de diversidade não é novo e foi percebido por outros pesquisadores. Yuan, Wang e Han (2014) propuseram o conceito de cobertura de *shapelets* para excluir *shapelets* redundantes, porém não abordamos este trabalho pois ele necessita que a medida de qualidade possua um ponto de corte, mas de qualquer forma é uma solução elegante e este problema merece maior pesquisa. Yuan, Wang e Han (2014) também ofereceram uma solução engenhosa ao problema de determinar a quantidade de atributos a ser utilizada na transformada, que também utiliza o conceito de cobertura, que é um parâmetro de importância e que por nós foi determinado de modo empírico. Não obstante, recomendamos que a amostragem aleatória também seja a *baseline* para influir diversidade nos atributos, mas acreditamos que uma solução satisfatória para o problema de diversidade não será encontrada com uma abordagem simples de filtragem de atributos. Acreditamos que é preciso levar em conta uma interação entre os atributos.

Por fim, contrastamos diretamente o nosso modelo proposto (amostragem aleatória de 5% com uso da medida de qualidade *in-class transitions*) com o estado da arte (uso do algoritmo de estimação dos parâmetros *min* e *max* e das medidas de qualidade de ganho de informação e *f-statistic*). Em todos os cenários o nosso modelo apresentou maior acurácia na maioria dos conjuntos dados, no entanto, quando contrastado com o modelo estado da arte com uso do ganho de informação, esta diferença de acurácia não era estatisticamente significante.

REFERÊNCIAS

ABE, H.; YAMAGUCHI, T. Implementing an integrated time-series data mining environment-a case study of medical kdd on chronic hepatitis. In: **1st international conference on complex medical engineering (CME2005)**. [S.l.: s.n.], 2005. Citado na página 23.

ANDERSON, E. The Irises of the Gaspe Peninsula. **Bulletin of the American Iris Society**, v. 59, p. 2–5, 1935. Citado na página 40.

ANDROULAKIS, I.; WU, J.; VITOLO, J.; ROTH, C. Selecting maximally informative genes to enable temporal expression profiling analysis. **Proc. of Foundations of Systems Biology in Engineering**, 2005. Citado na página 23.

APOSTOLICO, A.; BOCK, M. E.; LONARDI, S. Monotony of Surprise and Large-Scale Quest for Unusual Words. **Journal of Computational Biology**, v. 10, n. 3-4, p. 283–311, 2003. Citado na página 35.

BATISTA, G. E.; KEOGH, E. J.; TATAW, O. M.; SOUZA, V. M. de. CID: an efficient complexity-invariant distance for time series. **Data Mining and Knowledge Discovery**, v. 28, n. 3, p. 634–669, 2014. Citado na página 31.

BREIMAN, L. Random Forests. **Machine learning**, v. 45, n. 1, p. 5–32, 2001. Citado na página 44.

CELLY, B.; ZORDAN, V. Animated people textures. **Proc. of 17th International Conference on Computer Animation and Social Agents**, p. 331–338, 2004. Citado na página 23.

CHEN, Y.; KEOGH, E.; HU, B.; BEGUM, N.; BAGNALL, A.; MUEEN, A.; BATISTA, G. **The UCR Time Series Classification Archive**. 2015. <www.cs.ucr.edu/~eamonn/time_series_data/>. Citado 5 vezes nas páginas 19, 52, 53, 63 e 64.

DING, H.; TRAJCEVSKI, G.; SCHEUERMANN, P.; WANG, X.; KEOGH, E. Querying and mining of time series data: experimental comparison of representations and distance measures. **Proceedings of the VLDB Endowment**, v. 1, n. 2, p. 1542–1552, 2008. Citado 2 vezes nas páginas 24 e 39.

FISHER, R. A. The use of multiple measurements in taxonomic problems. **Annals of eugenics**, v. 7, n. 2, p. 179–188, set. 1936. Citado na página 40.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. The elements of statistical learning: Data mining, inference, and prediction. **Springer Series in Statistics**, New York, NY: Springer-Verlag New York, 2009. Citado 3 vezes nas páginas 44, 45 e 46.

FUAD, M. M. M.; MARTEAU, P. Towards a Faster Symbolic Aggregate Approximation Method. In: **Proceedings of the Fifth International Conference on Software and Data Technologies (ICSOFT)**. [S.l.: s.n.], 2010. v. 1, p. 305–310. Citado na página 34.

GIUSTI, R.; BATISTA, G. E. An empirical comparison of dissimilarity measures for time series classification. In: IEEE. **Intelligent Systems (BRACIS), 2013 Brazilian Conference on**. [S.l.], 2013. p. 82–88. Citado 2 vezes nas páginas 30 e 31.

GORDON, D.; HENDLER, D.; ROKACH, L. Fast Randomized Model Generation for Shapelet-Based Time Series Classification. **arXiv preprint arXiv:1209.5038**, 2012. Citado 2 vezes nas páginas 54 e 55.

GRABOCKA, J.; SCHILLING, N.; WISTUBA, M.; SCHMIDT-THIEME, L. Learning time-series shapelets. In: ACM. **Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.], 2014. p. 392–401. Citado na página 62.

HILLS, J.; LINES, J.; BARANAUSKAS, E.; MAPP, J.; BAGNALL, A. Classification of time series by shapelet transformation. **Data Mining and Knowledge Discovery**, v. 28, n. 4, p. 851–881, 2014. Citado 16 vezes nas páginas 25, 26, 27, 34, 37, 39, 56, 58, 59, 60, 61, 62, 65, 67, 77 e 78.

KAMATH, C.; FAN, Y. J. Finding motifs in wind generation time series data. In: IEEE. **Machine Learning and Applications (ICMLA), 2012 11th International Conference on**. [S.l.], 2012. v. 2, p. 481–486. Citado na página 23.

KEOGH, E. Exact indexing of dynamic time warping. In: VLDB Endowment. **Proceedings of the 28th international conference on Very Large Data Bases**. [S.l.], 2002. p. 406–417. Citado na página 30.

KEOGH, E.; CHAKRABARTI, K.; PAZZANI, M.; MEHROTRA, S. Dimensionality reduction for fast similarity search in large time series databases. **Knowledge and information Systems**, v. 3, n. 3, p. 263–286, 2001. Citado na página 34.

KEOGH, E.; KASETTY, S. On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. **Data Mining and Knowledge Discovery**, v. 7, n. 4, p. 349–371, 2003. Citado 2 vezes nas páginas 30 e 31.

LIN, J.; KEOGH, E.; WEI, L.; LONARDI, S. Experiencing SAX: a novel symbolic representation of time series. **Data Mining and Knowledge Discovery**, v. 15, n. 2, p. 107–144, 2007. Citado na página 34.

LINES, J.; BAGNALL, A. Alternative quality measures for time series shapelets. In: **Intelligent Data Engineering and Automated Learning-IDEAL 2012**. [S.l.]: Springer, 2012. p. 475–483. Citado 2 vezes nas páginas 25 e 26.

LINES, J.; DAVIS, L. M.; HILLS, J.; BAGNALL, A. A shapelet transform for time series classification. In: ACM. **Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.], 2012. p. 289–297. Citado 8 vezes nas páginas 23, 24, 25, 29, 39, 58, 59 e 65.

MCGOVERN, A.; ROSENDAHL, D.; KRUGER, A.; BEATON, M.; BROWN, R.; DROEGEMEIER, K. Understanding the formation of tornadoes through data mining. In: **5th conference on artificial intelligence and its applications to environmental sciences at the American meteorological society**. [S.l.: s.n.], 2007. Citado na página 23.

MUEEN, A. Enumeration of Time Series Motifs of All Lengths. In: IEEE. **Data Mining (ICDM), 2013 IEEE 13th International Conference on**. [S.l.], 2013. p. 547–556. Citado na página 54.

MUEEN, A.; KEOGH, E.; YOUNG, N. Logical-shapelets: an expressive primitive for time series classification. In: ACM. **Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.], 2011. p. 1154–1162. Citado 4 vezes nas páginas 31, 32, 54 e 55.

MUEEN, A.; NATH, S.; LIU, J. Fast approximate correlation for massive time-series data. In: ACM. **Proceedings of the 2010 ACM SIGMOD International Conference on Management of data**. [S.l.], 2010. p. 171–182. Citado na página 54.

OSHIRO, T. M.; PEREZ, P. S.; BARANAUSKAS, J. A. How Many Trees in a Random Forest? In: SPRINGER. **Machine Learning and Data Mining in Pattern Recognition**. [S.l.], 2012. p. 154–168. Citado na página 45.

PATRI, O. P.; SHARMA, A. B.; CHEN, H.; JIANG, G.; PANANGADAN, A. V.; PRASANNA, V. K. Extracting discriminative shapelets from heterogeneous sensor data. In: IEEE. **Big Data (Big Data), 2014 IEEE International Conference on**. [S.l.], 2014. p. 1095–1104. Citado na página 23.

QUINLAN, R. J. C4. 5: Programs for machine learning. Morgan Kaufmann, 1992. Citado na página 41.

RAKTHANMANON, T.; KEOGH, E. Fast shapelets: A scalable algorithm for discovering time series shapelets. In: SIAM. **International Conference on Data Mining (SDM)**. [S.l.], 2013. p. 668–676. Citado 2 vezes nas páginas 29 e 54.

RATANAMAHATANA, C. A.; KEOGH, E. Everything you know about dynamic time warping is wrong. In: CITESEER. **Third Workshop on Mining Temporal and Sequential Data**. [S.l.], 2004. Citado na página 30.

RATANAMAHATANA, C. A.; KEOGH, E. J. Making Time-series Classification More Accurate Using Learned Constraints. In: SIAM. **International Conference on Data Mining (SDM)**. [S.l.], 2004. p. 11–22. Citado na página 32.

VAPNIK, V. **Nature of Learning Theory**. [S.l.]: Springer Verlag, 1996. Citado na página 45.

WISTUBA, M.; GRABOCKA, J.; SCHMIDT-THIEME, L. Ultra-Fast Shapelets for Time Series Classification. **arXiv preprint arXiv:1503.05018**, 2015. Citado na página 61.

YE, L.; KEOGH, E. Time series shapelets: a new primitive for data mining. In: ACM. **Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.], 2009. p. 947–956. Citado 7 vezes nas páginas 23, 24, 32, 39, 41, 51 e 52.

YE, L.; WANG, X.; YANKOV, D.; KEOGH, E. J. The Asymmetric Approximate Anytime Join: A New Primitive with Applications to Data Mining. In: SIAM. **International Conference on Data Mining (SDM)**. [S.l.], 2008. p. 363–374. Citado na página 23.

YUAN, J.; WANG, Z.; HAN, M. A discriminative shapelets transformation for time series classification. **International Journal of Pattern Recognition and Artificial Intelligence**, v. 28, n. 06, p. 1450014, 2014. Citado 2 vezes nas páginas 61 e 78.