UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE INFORMÁTICA - CURSO DE CIÊNCIA DA COMPUTAÇÃO

Guilherme Lazzarotto de Lima

Classificação Automática de Batidas de Eletrocardiogramas

Monografia apresentada para obtenção do Grau de Bacharel em Ciência da Computação pela Universidade Federal do Rio Grande do Sul

Trabalho de Graduação

Orientador: Valter Roesler

Resumo

Este trabalho apresenta uma análise comparativa de métodos de classificação de batidas de eletrocardiograma (ECG) através do reconhecimento de padrões nos seus sinais. Essa análise visa identificar qual dentre os métodos existentes é a melhor opção, ou seja, a opção com maiores acurácia e sensitividades, para a utilização em sistemas de suporte ao diagnóstico de cardiopatias. Foram implementados os três principais métodos de classificação existentes e ainda um novo método, cujo desenvolvimento foi auxiliado por este trabalho. As implementações foram realizadas utilizando a linguagem de programação Java. Foram utilizadas mais de 11 mil gravações de ECG obtidas de um banco de dados de referência na área para validar o estudo.

A acurácia obtida pelo novo método proposto foi de 99,83% e as taxas de sensibilidade foram de no mínimo 99,48% para todos os tipos de batidas, o que supera os resultados dos outros métodos de classificação de batidas de ECG.

Abstract

This job presents a comparative analysis Electrocardiogram Beat Classification Methods using pattern recognition in ECG images. This analysis aims to identify which method is the better option, it means the most acurate and with greater sensitivities rates option, to be used on a diagnosis support system. The three main existing methods were implemented together with a new method, which development were helped by this job. The implementations were made using the Java programming language. More than 11 thousand ECG beats records from this area reference database were used to validate this study.

The accuracy obtained by the proposed method was 99.83% and the minimum sensitivities rates were 99.48% for all beat types, which is higher than the results obtained by the other ECG beat classification methods.

Dedicatória

Dedico este trabalho a Letícia Y Castro que tanto me apoiou.

Agradecimentos

Agradeço ao meu orientador Valter Roesler pela orientação e à Fernando Arena Varella pela parceria.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Neste capítulo são apresentados o objetivo, a motivação e a estrutura deste trabalho.

1.1 Motivação

A saúde pública brasileira é caracterizada por baixa disponibilidade de leitos e alta demanda de pacientes, o indicador de leitos por 1000 habitantes disponíveis ao SUS (Sistema Único de Saúde) apresenta um índice de 1,6 para o Brasil, 1,5 para a região Norte do país e um máximo de 1,9 para a região Sul, índice que não corresponde aos de 2,5 leitos para cada 1000 habitantes estabelecido pelo Ministério da Saúde [1], [2], [3]. Isso significa que existem muitas pessoas necessitando de atendimento porém a rede pública não é capaz de atender à todos. Além da já conhecida consequência deste quadro, filas de espera grandes e atendimentos realizados em espaços inapropriados, essas duas características implicam, comumente, na saída precoce de pacientes dos hospitais. Fato que ocorre pois o paciente acaba retornando às suas atividades, sem estar completamente restabelecido, para poder deixar seu leito para outro paciente que aparentemente necessita de maiores cuidados médicos.

Uma forma de resolver este problema é o uso do homecare. Homecare é uma palavra oriunda da língua inglesa, onde *care* significaria cuidado e *home* domicílio, enfim homecare é cuidado em domicílio. Como o nome sugere, o homecare é uma alternativa aos cuidados em hospitais, ao invés de ser internado em um hospital, o paciente recebe cuidados em casa, com mais privacidade, e ainda reduzindo gastos públicos com internação, bem como reduzindo a lotação de hospitais.

Segundo as estimativas da Organização Mundial de Saúde (WORLD HEALTH ORGANIZATION) sobre mortalidade e fardo de doenças para o ano de 2002, mais de um milhão de mortes causadas por doenças no Brasil, deste total quase 400 mil mortes foram causadas por doenças cardiovasculares, correspondendo à 32% das mortes. O estudo considera ainda outros 20 tipos de doenças, assim o fato de um tipo de doença causar mais de 30% das mortes é muito

significante [4].

Uma característica importante das cardiopatias, doenças cardiovasculares, é a necessidade do coração do paciente ser monitorado constantemente, em casos simples, medianos ou graves. Geralmente o monitoramento é feito por um Eletrocardiograma. O Eletrocardiograma (ECG) é um procedimento que registra a atividade elétrica do coração. Há mais de 80 anos, o ECG é utilizado como a base de diagnóstico de cardiopatias. O ECG é um procedimento não-invasivo, simples e que pode ser aplicado constantemente, além disso é um procedimento barato e extremamente versátil. Sua versatilidade se deve pois ele é utilizado tanto no diagnóstico de arritmias e isquemias, quanto a outras que afetam o coração direta ou indiretamente [5].

Os avanços tecnológicos atuais permitem que se faça o monitoramento do paciente remotamente, ou seja, os sinais do paciente podem ser monitorados mais facilmente e constantemente, sem atrapalhar sua rotina. O uso de tecnologias de telecomunicações na medicina, a telemedicina, aliada ao Homecare, caracteriza o telehomecare [6]. Enfim, o contexto da saúde pública brasileira demonstra que necessita-se cada vez mais de alternativas de atendimento e uma boa alternativa é a utilização do telehomecare, para no caso das cardiopatias monitorar os sinais cardíacos de um paciente via ECG.

1.2 Objetivos

Existem muitos métodos de classificação automática de ECG, isto é, reconhecer cardiopatias em um registro de ECG. Este trabalho tem dois grandes objetivos, o primeiro é realizar uma análise comparativa dos principais métodos de classificação existentes, identificando as vantagens e desvantagens dos principais métodos existentes. Pretende-se realizar a implementação dos melhores métodos dentre os analisados para verificar se na prática os resultados obtidos correspondem aos resultados relatados nas publicações dos métodos.

Além disso, o segundo objetivo deste trabalho é, baseado na análise teórica e prática dos métodos, estabelecer melhorias para os métodos. Nesta direção, este trabalho auxiliou o desenvolvimento de um novo método de classificação de batidas de ECG, desenvolvido no mestrado de Fernando Arena Varella, intitulado Método de Varella e Lima. Este novo método é utilizado neste trabalho para evidenciar seu desempenho e confrontar seus resultados com os métodos selecionados para implementação.

1.3 Estrutura

Este trabalho organiza-se da seguinte forma, primeiramente no capítulo 2 serão discutidos os conceitos básicos necessários para a compreensão da tarefa de classificação automática de ECG. A seguir no capítulo 3 é apresentada a base teórica necessária para a implementação dos principais métodos existentes. Em seguida, no capítulo 4 serão apresentadas as características dos principais métodos de classificação de ECG existentes, bem como a análise comparativa dos mesmos, além da descrição do método de Varella e Lima. No capítulo 5 será apresentado o projeto de implementação dos métodos escolhidos. Enquanto que no capítulo 6 serão descritos os detalhes da implementação dos mesmos. Já no capítulo 7 será descrito o ambiente de testes da implementação e serão apresentados os resultados dos testes realizados. Finalmente, o capítulo 8 finaliza o trabalho com conclusões e considerações finais.

2 *Conceitos Básicos de Cardiologia*

Neste capítulo são apresentados os conceitos fundamentais de cardiologia para a interpretação de eletrocardiogramas, são apresentados os padrões do sinal que correspondem a uma determinada cardiopatia.

Basicamente um eletrocardiograma é um exame médico que realiza um registro da atividade elétrica do coração. O fato do coração apresentar uma atividade elétrica é vital, pois é através desta atividade que o sangue é impulsionado para os pulmões, onde ocorrem a troca entre gás carbônico e oxigênio, e para todo o resto do corpo, que necessita de oxigênio, simultaneamente.

2.1 O Ciclo Cardíaco

A atividade elétrica cardíaca acontece da seguinte forma. Primeiramente o coração encontra-se em um estado de repouso. Neste estado suas células encontram se polarizadas, com seu interior carregado negativamente. Quando estimuladas, estas células despolarizam-se, contraem e ficam então carregadas positivamente.

Localizado no átrio direito, encontra-se o Nódulo Sino-Atrial (Nódulo SA) que é o responsável pela contração do coração. Ele desencadeia uma propagação de despolarização (cargas positivas), estimulando ambos os átrios e posteriormente produz uma contração simultânea dos átrios, esta fase é conhecida como **sístole atrial**. Está contração impulsiona o sangue na direção dos ventrículos. Então a onda de despolarização atinge o Nódulo Atrioventricular (Nódulo AV), situado entre os átrios, que após uma pausa de 1/10 segundos, necessários para que o sangue entre nos ventrículos, transmite o estimulo aos ventrículos através do Feixe Atrioventricular (ou Feixe de His).

O Feixe de His é divido em dois ramos, o Ramo Direito que estende-se por todo o Ventrículo Direito e o Ramo Esquerdo que por sua vez estende-se pelo Ventrículo Esquerdo. Ambos os ramos possuem ramificações, as chamadas fibras de Purkinje, são essas fibras que realmente transmitem o estimulo elétrico despolarizante para as células miocárdicas. Consequentemente

os ventrículos são despolarizados simultaneamente e com isso contraem, fase conhecida como **sístole ventricular**. Após uma certa pausa, os ventrículos repolarizam-se, esta fase é conhecida como fase de **repouso entre os batimentos**. A figura 2.1 demonstra este ciclo.

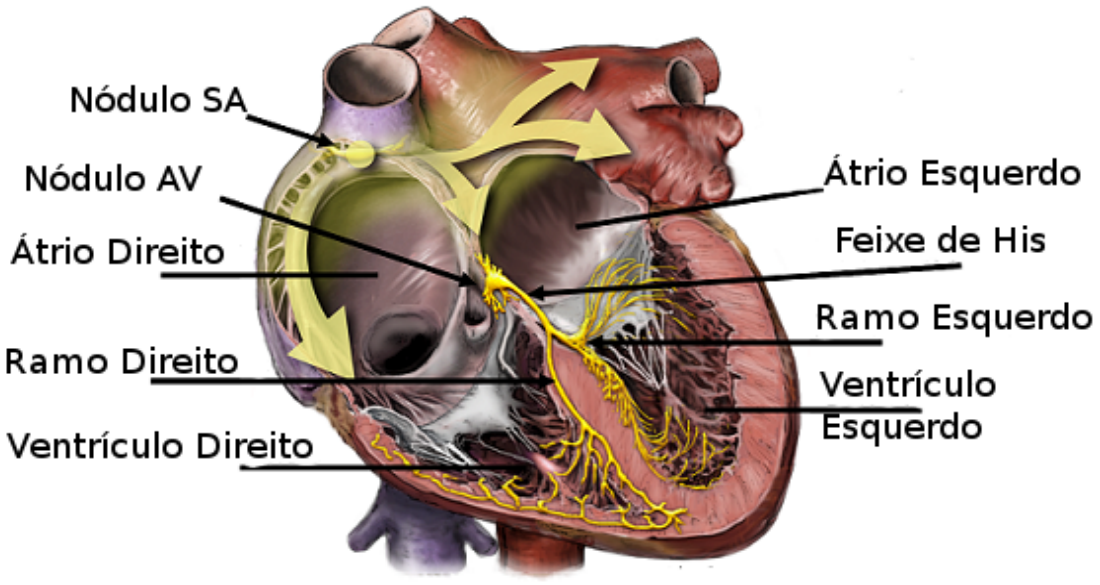

Figura 2.1: Coração e sentido da despolarização. Adaptada de [7].

2.2 Funcionamento do Eletrocardiograma

O ciclo cardíaco é captado pelo eletrocardiograma com o auxilio de eletrodos cutâneos. Os eletrodos são colocados aos pares no corpo do paciente. Um par de eletrodos representa uma derivação, um ECG padrão possui doze derivações, seis derivações precordiais e seis derivações periféricas. Cada derivação é formada por um eletrodo positivo e um eletrodo negativo.

Quando uma onda positiva de despolarização dentro das células cardíacas se move na direção de um eletrodo positivo instalado na pele do paciente, registra-se uma deflexão positiva no ECG. Para obter as derivações periféricas ou dos membros, coloca-se eletrodos no braço direito e esquerdo e na perna esquerda, disposição que forma um triângulo, como verifica-se na figura 2.2. A Derivação I é composta por um par de eletrodos dispostos horizontalmente, o eletrodo negativo é colocado no braço direito e o positivo no braço esquerdo. Na Derivação II o eletrodo negativo também é colocado no braço direito, porém o eletrodo positivo é colocado na perna esquerda. E finalmente tem-se a Derivação III na qual novamente tem-se um eletrodo positivo na perna esquerda e um eletrodo negativo no braço esquerdo.

As outras três derivações periféricas são a AVR, AVF e AVL. A AVR utiliza um eletrodo

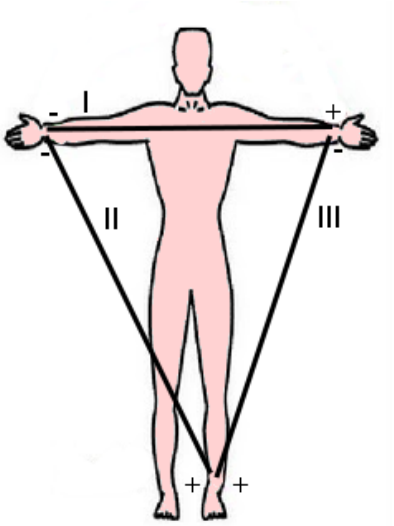

Figura 2.2: Derivações Periféricas: I, II e III

positivo no braço direito e um eletrodo negativo comum (fio terra) no braço esquerdo, pé esquerdo e pé direito. A AVL é semelhante, porém eletrodo positivo no braço esquerdo e um eletrodo negativo comum no braço direito, pé esquerdo e pé direito. Já a AVF é obtida por um eletrodo positivo no pé esquerdo e um eletrodo negativo comum no braço esquerdo, braço direito e pé direito. Como pode ser visto na figura 2.3.

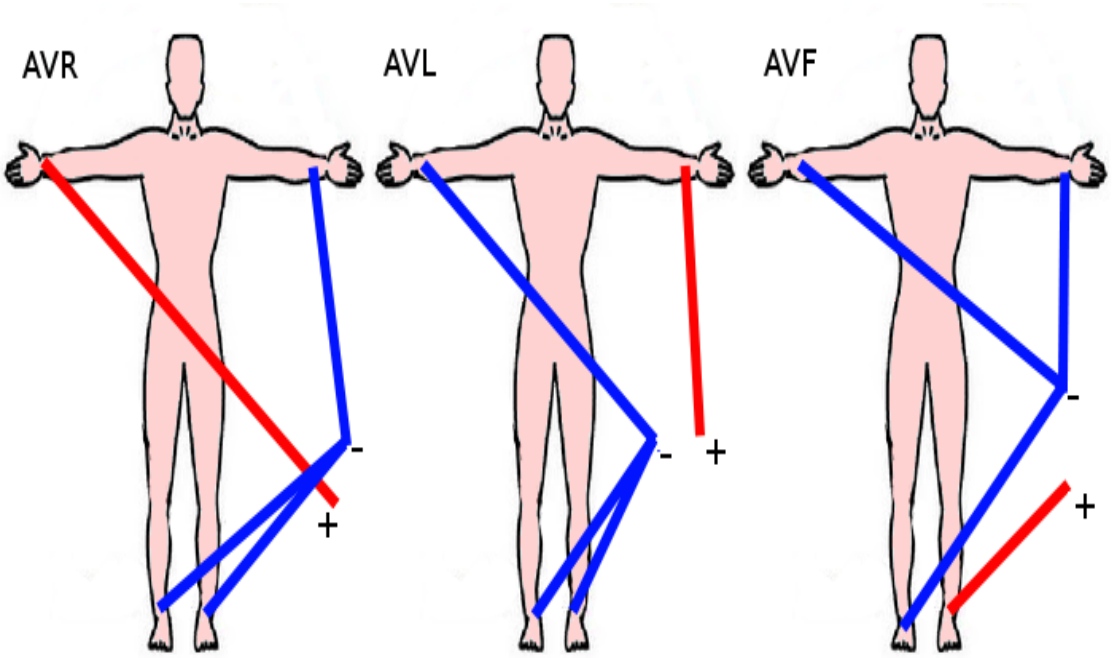

Figura 2.3: Derivações Periféricas: AVR, AVL e AVF

As derivações precordiais ou torácicas são obtidas de uma maneira relativamente diferente, para obtê-las coloca-se eletrodos positivos no tórax do paciente, nas posições demonstradas na figura 2.4. Com o intuito de cobrir totalmente o coração do paciente em sua posição anatômica

dentro do tórax.

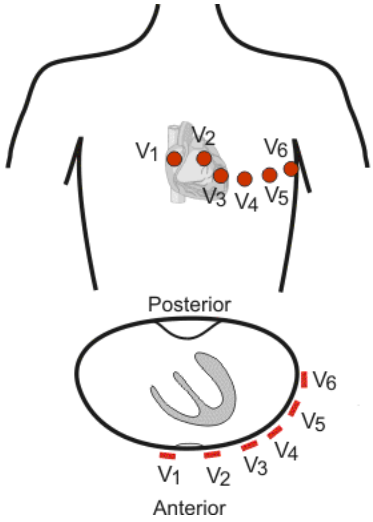

Figura 2.4: Derivações Precordiais: V1, V2, V3, V4, V5 e V6. Retirado de [8]

Os eletrodos colocados no tórax do paciente são todos positivos, assim estas derivações consideram o dorso do paciente como o pólo negativo da derivação. Como notam-se na figura 2.4 as derivações V1 e V2 estão localizadas no lado direito do coração, V3 e V4 localizam-se sobre o septo interventricular e V5 e V6 ficam sobre o lado esquerdo do coração.

Finalmente, o registro do eletrocardiograma acontece da seguinte forma, quando a onda de despolarização se aproxima do eletrodo positivo, evidenciam-se uma deflexão para cima no traçado do ECG. Enquanto que quando a onda de despolarização se afasta do eletrodo positivo, evidenciam-se uma deflexão para baixo. Com isso obtêm-se o registro da atividade elétrica.

2.3 O Registro do ECG

Na figura 2.5 observa-se uma batida típica que é registrada em uma derivação de um eletrocardiograma. Ela contém 5 ondas, a onda P, Q, R, S e T. A onda P é a deflexão causada pela contração atrial. Já as ondas Q, R e S formam o chamado Complexo QRS e representam a contração ventricular. Por fim, existe a onda T que representa a Repolarização Ventricular. Além disso a figura contém o chamado segmento PR que é uma pausa existente entre a onda P e Q, que representa a pausa de 1/10 segundos antes do Nódulo AV ser estimulado verdadeiramente. O segmento ST por sua vez representa a pausa existente entre a contração ventricular e a repolarização ventricular. Ainda existe o intervalo PR que indica a o tempo que a onda levou para chegar dos átrios aos ventrículos e o intervalo QT que indica o tempo entre a contração ventricular e a repolarização ventricular.

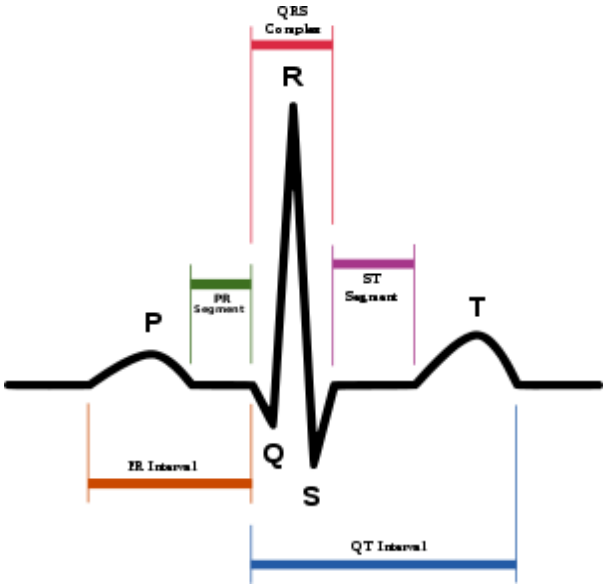

Figura 2.5: Batida típica em um Eletrocardiograma. Retirado de [9]

2.4 Interpretação do ECG

A partir de um ECG pode-se interpretar a Frequência, Ritmo, Eixo, Hipertrofia e Infarto. Determina-se a frequência que os os batimentos cardíacos ocorrem, se as batidas estão reguladas em um ritmo. Além disso, identifica-se alterações neste ritmo, as chamadas arritmias. Ainda identifica-se a direção da despolarização que causa a contração das células cardíacas, bem como possíveis anormalidades. Também analisa-se a existência de hipertrofias cardíacas, ou seja, aumento da massa muscular do coração. E finalmente, analisa-se a ocorrência de Infarto, isto é, a oclusão de uma artéria coronária que serve o ventrículo esquerdo. Entretanto, os métodos analisados neste trabalho apenas classificam arritmias, sendo necessário portanto apenas a interpretação da frequência cardíaca e do ritmo cardíaco em um ECG.

No que se refere a frequência cardíaca, é importante saber que no estado normal, o Nódulo SA determina a frequência cardíaca. Localizado no átrio direito, o Nódulo SA, é o marca-passo cardíaco normal. Porém em situações anormais, outras partes do coração podem determinar a frequência cardíaca, estas partes são conhecidas como marca-passos ectópicos. Os marca-passos ectópicos localizam em diversas partes do coração, inclusive nos átrios, ventrículos e nódulo AV.

Na figura 2.6 faz-se uma estimativa da frequência cardíaca a partir de um ECG. Para extrair a frequência de um ECG primeiramente deve-se escolher uma onda R que coincida com uma linha mais escura do papel milimetrado, a linha escura seguinte corresponde à uma frequência de 300 ciclos/min, a seguinte 150 ciclos/min e em seguida 100 ciclos/min. A próxima linha

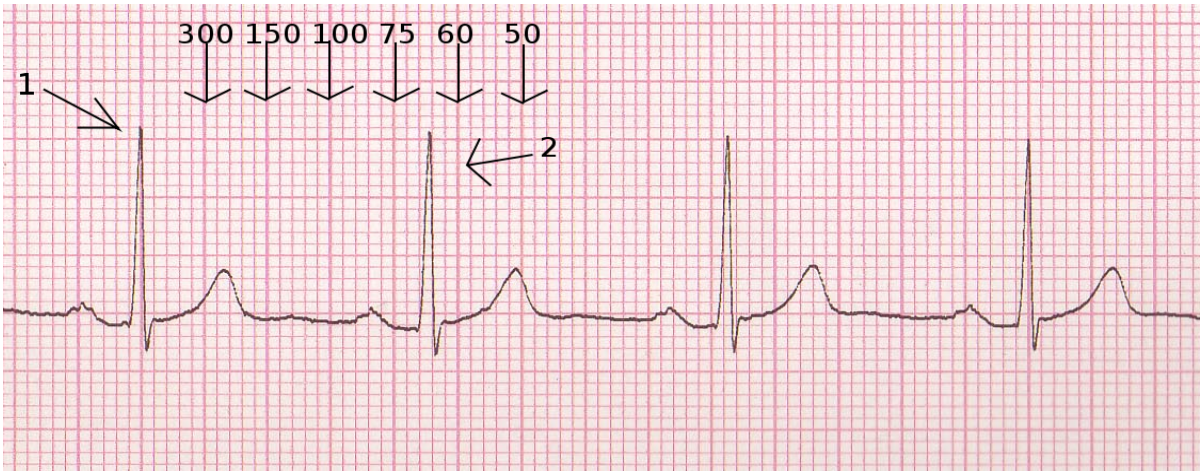

Figura 2.6: Cálculo da Frequência Cardíaca.Adaptado de [10]

escura equivale à 75 ciclos/min, a subsequente à 60 e a sexta à 50. O lugar onde a próxima onda R está determina a frequência. No exemplo tem uma frequência entre 75 e 60 ciclos/min, portanto, uma frequência normal.

A frequência normal é uma frequência entre 60 e 100 ciclos por segundo, sendo que frequência superior à 100/min caracteriza Taquicardia Sinusal e frequência inferior à 60/min caracteriza uma Bradicardia Sinusal. Em caso de patologias ou emergências os marca-passos ectópicos estabelecem um ritmo próprio, marca-passos ectópicos localizados nos átrios possuem frequência de 75/min, no nódulo AV de 60/min e nos ventrículos de 30-40/min. Em alguns casos o nódulo AV ou o ventrículo, podem disparar frequências rápidas de 150-250/min.

Além da análise da frequência cardíaca, será fundamental para classificar as batidas do ECG, ter noções das características das cardiopatias relacionadas ao ritmo cardíaco, bem como saber como as identificá-las em um ECG. Primeiramente, um Ritmo Normal ou Regular apresenta distâncias iguais entre ondas semelhantes, este ritmo geralmente é determinado pelo Nódulo SA portanto é chamado de Ritmo Sinusal Regular ou Ritmo Sinusal Normal. As arritmias, cardiopatias relacionadas ao ritmo cardíaco, elas podem ser de quatro tipos gerais: **Ritmo Variável, Extrassístoles, Ritmos rápidos e Bloqueios**.

As três arritmias de ritmo variável formam um grupo de ritmos irregulares em sequência normal de ondas (P-QRS-T), com alterações contínuas no ritmo, conforme visualiza-se na figura 2.7.

Primeiro existe a **Arritmia Sinusal** causada por um comportamento irregular do Nódulo SA. Identificada no ECG por um ritmo irregular, ou seja, distâncias diferentes entre ondas idênticas e ondas P idênticas. Ela pode indicar uma doença relacionada a artéria coronária. Outra arritmia de ritmo variável chama-se **Marca-passo Migratório** neste caso marca-passos

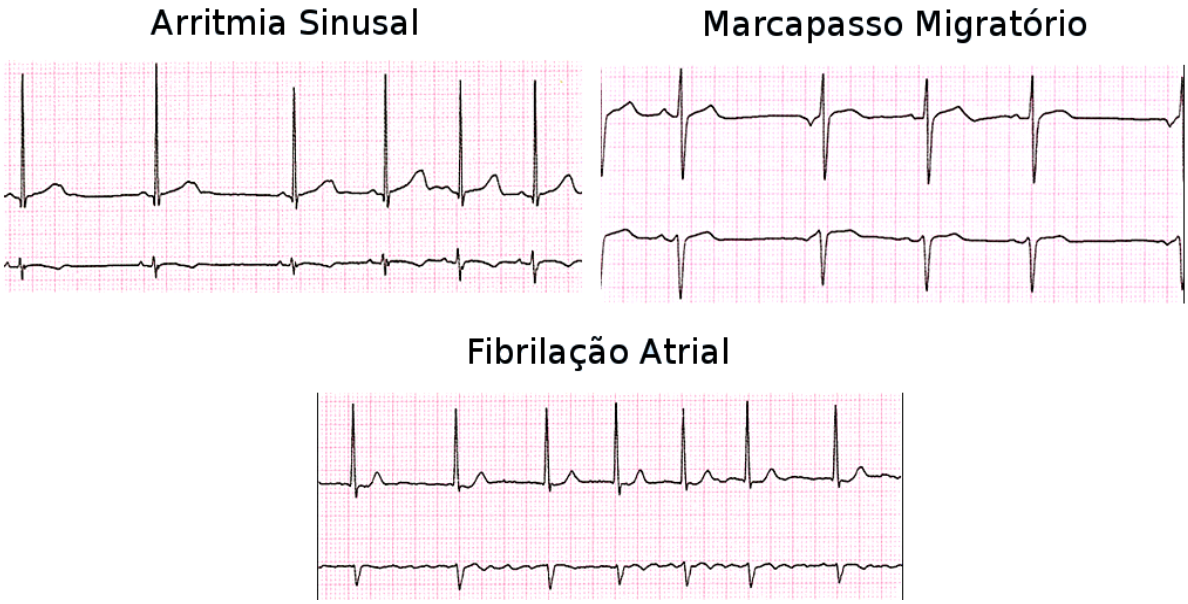

Figura 2.7: Arritmia Sinusal, Marcapasso Migratório e Fibrilação Atrial. Adaptado de [11]

ectópicos disparam de forma aleatória. O que ocasiona novamente um ritmo irregular, porém desta vez as ondas P mudam de forma, conforme a sede do Marca-passo muda. E finalmente existe a **Fibrilação Atrial**, causada por descarga de marca-passos atriais múltiplos. O ritmo observado é irregular e não existem ondas P reais, apenas múltiplas deflexões atriais ectópicas.

O grupo das extra-sístoles é composto extra-sístoles e batimentos de escape. A **Extrassístole Atrial** acontece quando um marcapasso atrial ectópico dispara uma onda P precocemente, esta onda é seguida de um complexo QRS normal. A **Extrassístole Nodal** por sua vez ocorre quando um marca-passo no nódulo AV dispara e acarreta em um complexo QRS não precedido de onda P. Já a **Extrassístole Ventricular** ocorre em decorrência de um marcapasso ventricular e é caracterizada por um complexo QRS grande, largo e precoce, seguido de uma pausa compensadora. A figura 2.8 demonstra as extra-sístoles comentadas.

Os batimentos de escape no entanto ocorrem após uma pausa compensadora correspondente a mais de um ciclo completo. O **Escape Atrial** parece uma Extrassístole Atrial porém é seguido de uma pausa compensadora. O **Escapes Nodal e Ventricular** parecem Extrassístoles Nodal e Ventricular, respectivamente, entretanto também são seguidos de uma pausa compensadora. A figura 2.9 exibe as batidas de escape.

Outro grupo de arritmias é compostos pelas arritmias de ritmo rápido, que é subdividido em Taquicardias Paroxísticas, Flutters e Fibrilações. A **Taquicardia Atrial Paroxística** é causada por descargas súbitas e rápidas de marcapassos ectópicos atriais, as ondas apresentam com isso uma sequência normal, com uma frequência de 150-250 ciclos/min e ainda ondas P podem não aparecer. Já a Taquicardia Nodal Paroxística é originada no nódulo AV, logo não há ondas

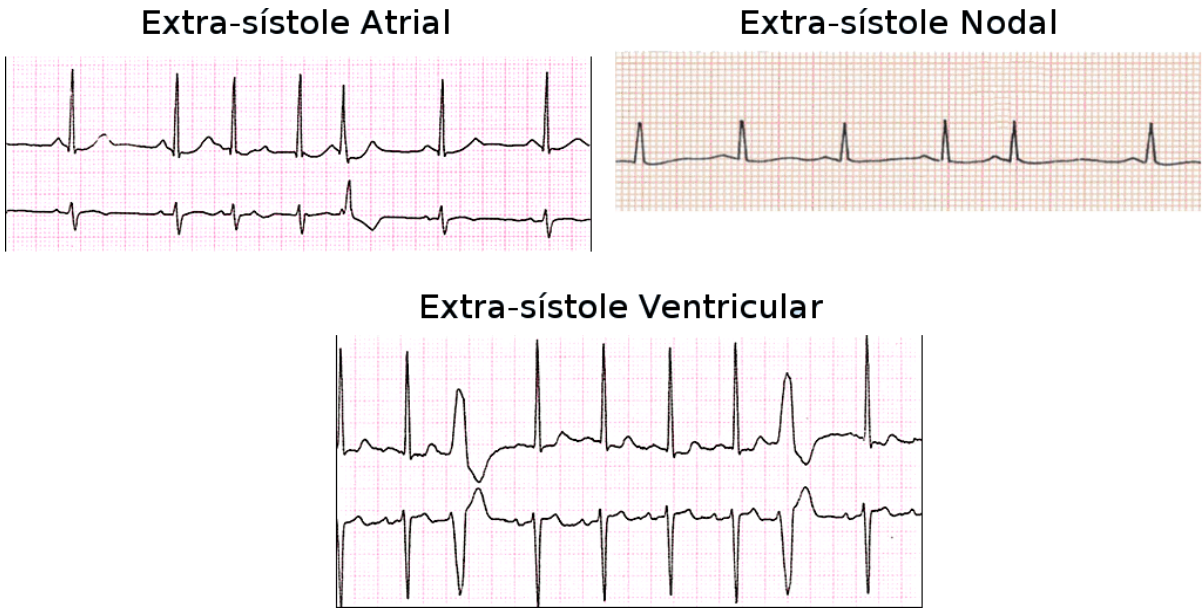

Figura 2.8: Extrassístoles

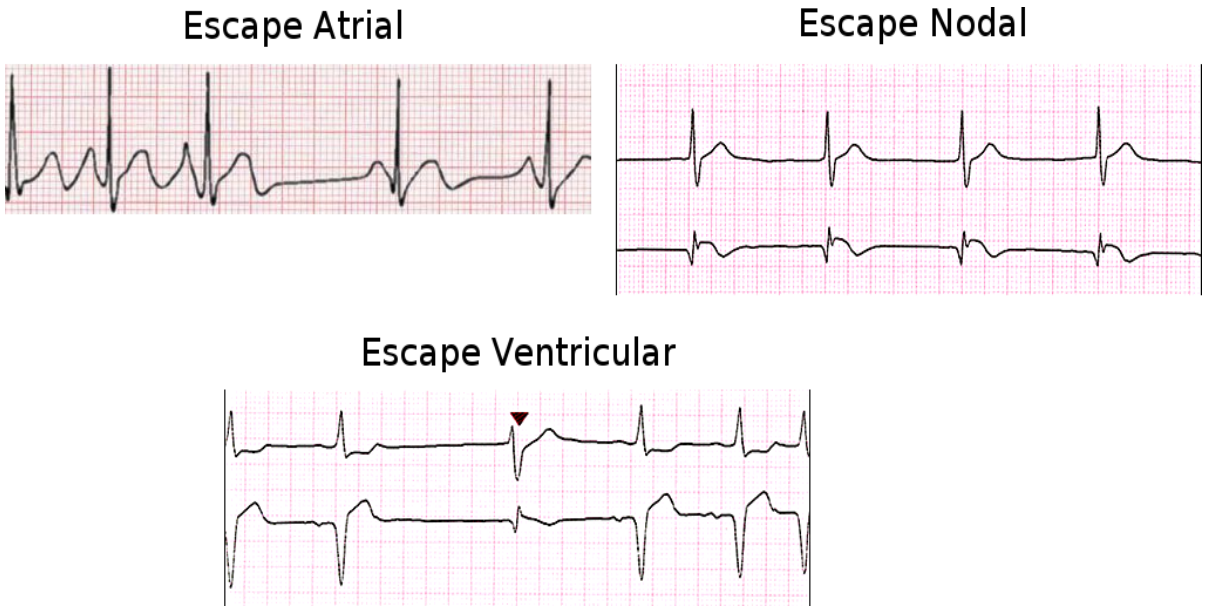

Figura 2.9: Batidas de Escape

P e a frequência é de 150-250 ciclos/min. Enfim existe a Taquicardia Ventricular Paroxística que é desencadeada por marcapassos ectópicos ventriculares e aparenta ser uma sucessão de E.S.V e novamente a frequência é de 150-250 ciclos/min. Na figura 2.10 são apresentadas as taquicardias paroxísticas.

O Flutter Atrial é causado por marcapassos ectópicos atriais que disparam com uma frequência de 250-300 ciclos/min. Caracteriza-se por um sucessão rápida e contínua de ondas P idênticas. Já o Flutter Ventricular é produzido por um único marca-passo ectópico ventricular que dispara à 250-300 ciclos/min, causando ondas difásicas regulares como uma série de

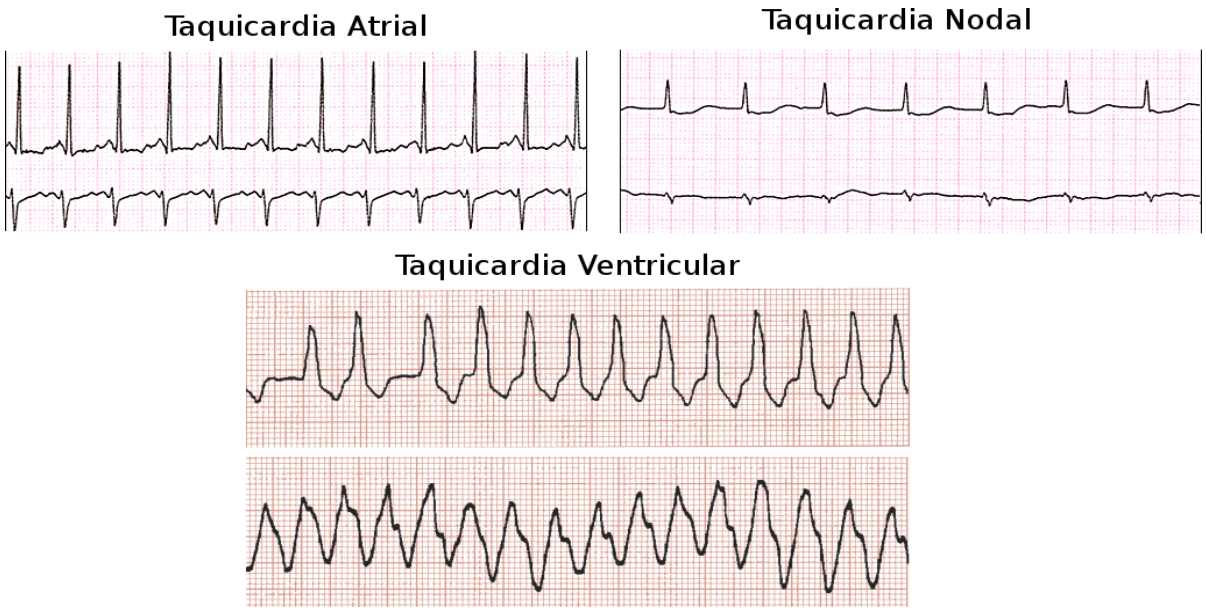

Figura 2.10: Taquicardias Paroxísticas

ondulações sinusoidais. Ainda, a Fibrilação Ventricular é causada por uma atividade elétrica totalmente anárquica produzida por estímulos ectópicos ventriculares que causam uma contração caótica do ventrículos. Possui um aspecto totalmente irregular e é mortal.

Finalmente existem as cardiopatias que fazem parte do grupo intitulado de Bloqueio Cardíaco. Estas cardiopatias podem ocorrer no Nódulo SA, Nódulo AV ou ainda nos Ramos do Feixe de His. São na verdade bloqueios elétricos que impedem a passagem de estímulos elétricos.

Primeiro existe o Bloqueio SA, é uma parada do Nódulo SA, ou seja, o Nódulo SA não desencadeia uma onda de despolarização. Ocasiona a ausência de pelo menos um ciclo no ECG, ausência sucedida pela retomada da sua atividade de estimulação. A figura 2.11 mostra o Bloqueio SA em um ECG.

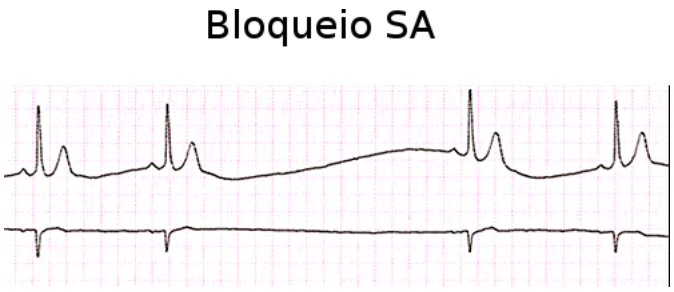

Figura 2.11: Bloqueio SA

O Bloqueio AV por sua vez pode ser de 1º, 2º e 3º grau. O Bloqueio AV de 1º Grau causa um retardo maior do que o normal para a total estimulação do Nódulo AV e posterior estimulação dos ventrículos, é identificado no ECG por um intervalo P-R maior que um quadrado

grande (0,2 segundos).

O Bloqueio de 2º Grau acontece quando são necessários dois ou mais impulsos atriais para estimular o Nódulo AV, aparece no ECG como duas ou mais ondas P precedendo um QRS. Relacionado ao Bloqueio 2º, existe o Fenômeno de Wenchebach que ocorre quando o intervalo P-R cresce gradualmente até que o nódulo AV não seja mais estimulado, causando ausência de QRS em um ECG. Este fenômeno é uma forma de Bloqueio 2º, chamado de Mobitz I. Além de Mobitz I, existe o Mobitz II que é caracterizado pela ausência de QRS mesmo sem um intervalo P-R alongado.

Ainda existe o Bloqueio 3º ou Bloqueio Completo que ocorre quando nenhum dos impulsos atriais consegue estimular o Nódulo AV, assim um marcapasso ectópico ventricular entra em ação e estimula os ventrículos. é identificado no ECG por frequência atrial (das ondas P) diferente da frequência ventricular (do complexo QRS). A figura 2.12 mostra os Bloqueios AV.

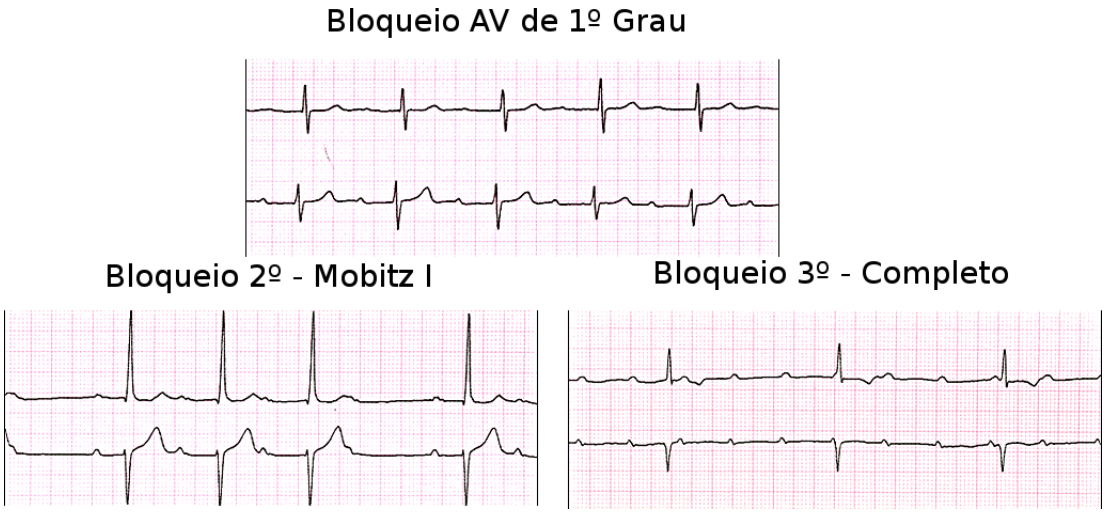

Figura 2.12: Bloqueios AV

Finalmente, existem os Bloqueios de Ramo que acontecem quando ocorre um retardamento na transmissão do impulso elétrico dos ramos para o ventrículo correspondente. A existência de um bloqueio de ramo é facilmente identificada pela presença de um complexo QRS alargado (com largura maior que três quadradinhos). O QRS largo nada mais é que a composição de um QRS originário do ramo desbloqueado e um QRS atrasado originado pelo ramo com bloqueio, nele percebe-se a existência de duas ondas R, a R e R' sucessivamente. Para saber qual ramo está bloqueado, verifica-se a existência de ondas R e R' e ondas S largas. Se existem ondas RR' em V1 e V2 e ondas S largas em V5 e V6, o bloqueio é de ramo Direito, senão se existem ondas RR' em V5 e V6 e ondas S largas em V1 e V2, o bloqueio é de Ramo Esquerdo. Na figura 2.13 visualiza-se uma amostra de ECG com Bloqueios de Ramo.

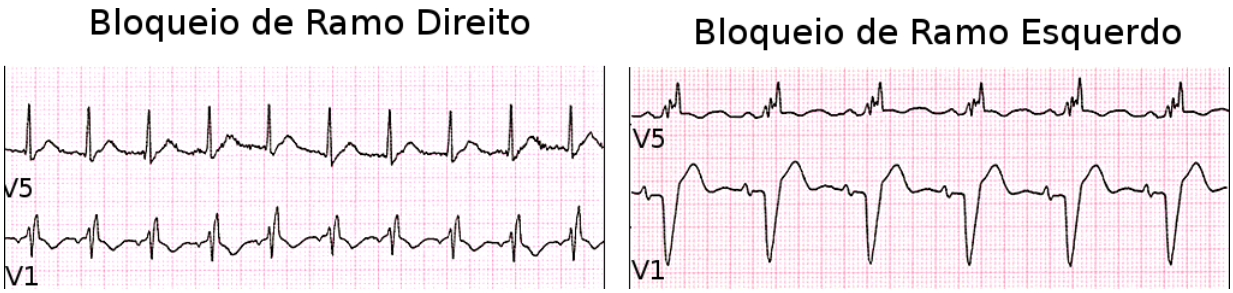

Figura 2.13: Bloqueios de Ramo

2.5 Resumo

O Ciclo Cardíaco é um fenômeno vital para os seres humanos, basicamente tem-se que o Nódulo SA desencadeia uma onda de despolarização, e consequente contração, nos átrios (sístole atrial) e posteriormente, com o auxílio do feixe de His,nos ventrículos (sístole ventricular). Após um repouso, átrios e ventrículos repolarizam-se podendo inciar novamente o ciclo. É através desta atividade que o sangue é impulsionado para os pulmões, onde ocorrem a troca entre gás carbônico e oxigênio, e para todo o resto do corpo, que necessita de oxigênio, simultaneamente.

O exame de ECG registra o comportamento desta atividade elétrica por meio de derivações precordiais e periféricas, formadas por um par de eletrodos cutâneos com cargas distintas. As derivações acabam por registrar deflexões (ondas) que representam cada uma das fases do ciclo cardíaco, geralmente tem-se 5 ondas, a onda P que é a deflexão causada pela contração atrial. Já as ondas Q, R e S formam o chamado Complexo QRS e representam a contração ventricular. Por fim, existe a onda T que representa a Repolarização Ventricular.

A partir de um registro de ECG é possível identificar-se várias características do coração bem como a Frequência, o Ritmo, o Eixo. Além disso, pode-se identificar alterações no ritmo, as chamadas arritmias, que podem ser classificadas como Arritmias de Ritmo Variável, Extrassístoles, Ritmos rápidos e Bloqueios.

3 *Base Teórica dos Métodos*

Neste capítulo serão introduzidos alguns preceitos básicos utilizados na maioria dos métodos de classificação de batidas de eletrocardiograma existentes, focando nas técnicas utilizadas nos três métodos que este trabalho implementa.

Essas técnicas são utilizadas para extrair características e/ou informações relevantes do sinal, ou seja, passíveis de classificação e ainda técnicas para a realização da classificação destas características. As seções que seguem apresentam as técnicas comumente utilizada nos principais métodos existentes.

3.1 Transformada Discreta de Ondaleta

As ondaletas são funções muito utilizadas no processamento de sinais, esta ampla utilização foi que incentivou o desenvolvimento desta área. Enquanto que a análise de Fourier trabalha apenas no domínio da frequência, a análise de ondaletas trabalha também no domínio do tempo, fato que propicia a análise de características locais do sinal, o que é muito importante para a classificação de batidas de ECG. Tanto a análise de Fourier quanto a análise de ondaletas pretendem aproximar um sinal, seja por uma combinação linear de senos e cossenos (análise de Fourier) ou ondaletas. A aproximação do sinal na análise de ondaletas é realizada por um processo chamado Transformada de Ondaleta que possui as variantes contínua e discreta.

A Transformada Contínua de Ondaleta (Continuos Wavelet Transform - CWT) de um sinal f é o produto interno do sinal por uma ondaleta, neste caso chamada de ondaleta-mãe ψ, como verifica-se na seguinte fórmula:

$$(W_\psi f)(a,b) = |a|^{-1/2} \int_{-\infty}^{\infty} f(t)\psi(\frac{t-b}{a})dt, a,b \in R, a \neq 0$$

A ondaleta-mãe ψ é dada pela seguinte fórmula:

$$\psi_{j,k}(t) = \psi(2^j t - k), j,k \in Z$$

A Transformada Discreta de Ondaleta (Discrete Wavelet Transform - DWT) de um sinal expresso na forma de um um vetor de amostras $X = (X_0, X_1, ..., X_T)'$ pode ser expressa como:

$$d_{j,k}^{(\psi)} = \sum_{t=0}^{T-1} X_t \psi_{j,k}(t/T)$$

Na prática, porém obtemos esta transformada utilizando um algoritmo piramidal, onde uma série de filtros passa-baixo e passa-alto são aplicados.

O Algoritmo Piramidal para calcular a transformada de ondaleta foi proposto por Stephane G. Mallat [12] é detalhado na figura 3.1 e demonstrado graficamente na figura 3.2.

Entrada: Um vetor X, um filtro passa-baixo L, um filtro passa-alto H e o número n de passos
Saída : Coeficientes de aproximação C_n e coeficientes de detalhe D_n

```
1 sinal ← sub-amostragem(X);
2 para i ← 1 to n faça
3 |   c_aux ← convolução((X,L));
4 |   d_aux ← convolução((X,H));
5 |   C_i ← sub-amostragem(c_aux));
6 |   D_i ← sub-amostragem(d_aux));
7 |   sinal ← C_i;
8 fim
```

Figura 3.1: Algoritmo Piramidal

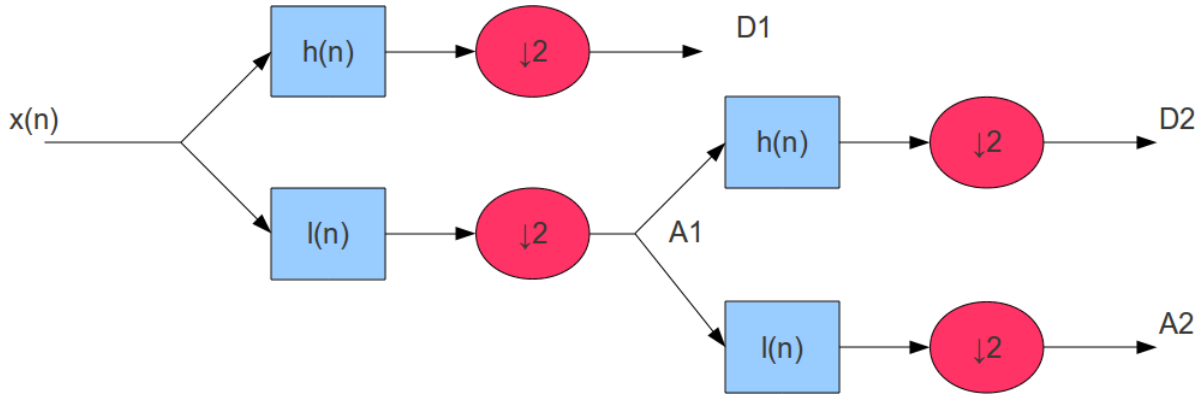

Figura 3.2: Esquema do Algoritmo Piramidal

O algoritmo utiliza filtros passa-baixo L e passa-alto H para calcular n coeficientes de aproximação e n coeficientes de detalhe, respectivamente. Após a convolução do sinal com um filtro, faz-se uma sub-amostragem do resultado. A saída sub-amostrada da convolução realizada pelo filtro passa-baixo pode ser reutilizada para uma nova iteração do algoritmo. Os filtros utilizados

dependem do tipo de ondaleta-mãe que desejamos utilizar para transformar o sinal. A Ondaleta mais comum é a ondaleta de Haar, os coeficientes dos filtros de passa-baixo e passa-alto são os seguintes:

$$H = [0.70710678118654757, 0.70710678118654757]$$

$$L = [-0.70710678118654757, 0.70710678118654757]$$

A representação gráfica desta ondaleta-mãe é demonstrada na figura 3.3.

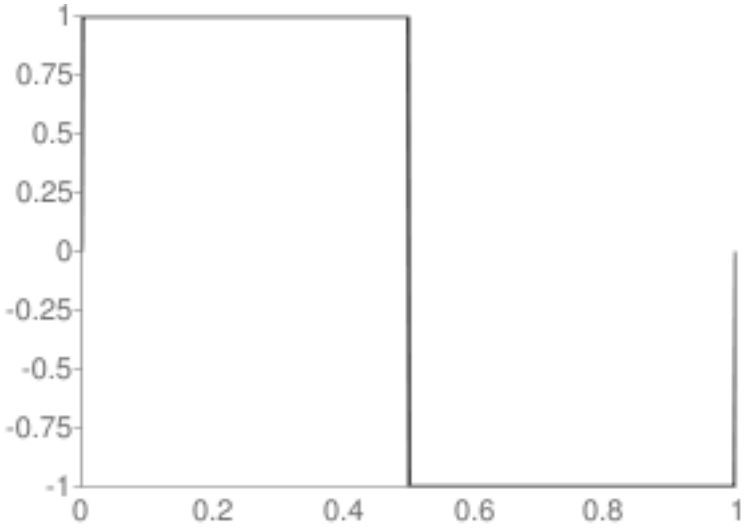

Figura 3.3: Ondaleta de Haar

Graças às sub-amostragens, a abordagem piramidal reduz a redundância de informação existente nos coeficientes. Entretanto sub-amostrar também reduz a resolução temporal dos coeficientes obtidos, ou seja, diminui-se a redundância dos coeficientes porém mesmo que o sinal seja menos redundante a precisão das medidas com respeito ao tempo também diminui. Portanto pode-se utilizar uma abordagem chamada algoritmo *à trous* que lida com este problema realizando a transformação sem sub-amostragem.

O algoritmo *à trous* utiliza também filtros passa-baixo L e passa-alto H para calcular n coeficientes de aproximação e n coeficientes de detalhe. O processo inicial de transformação é o mesmo do algoritmo piramidal, realiza-se a convolução do sinal com o filtro passa-baixas obtendo-se assim um conjunto de coeficientes de aproximação. Da forma semelhante, realiza-se a convolução do sinal com o filtro passa-altas obtendo coeficientes de detalhe do sinal. Aplicação dos filtros pode ser cascateada sobre os coeficientes de aproximação tantas vezes quanto se queira.

3.2 Métodos Estatísticos

Nesta seção serão descritos alguns métodos estatísticos utilizados nos métodos de classificação de batidas de ECG. A utilização de métodos estatísticos auxilia na obtenção de características significantes para a realização da classificação e/ou diferenciação dos diferentes padrões de sinal que correspondem a um determinado tipo de batida. Os métodos que este trabalho analisa utilizam estatísticas básicas, análise de componentes principais e estatística de ordem superior que serão abordadas nas subseções seguintes.

3.2.1 Métodos Estatísticos Básicos

Essa seção descreve métodos estatísticos básicos utilizados na tarefa de classificar batidas de ECG, são eles a variância, desvio padrão, autocorrelação e amplitude relativa.

A **variância** é definida como a média do quadrado da distância de cada valor até a média, na prática ela demonstra a diferença de um determinado valor de uma amostra em relação ao valor médio da amostra. Pode ser expressa pela seguinte fórmula:

$$\sigma_x^2 = \frac{1}{N}\sum_{n=1}^{N}[x(n)-\overline{x}]^2$$

A partir da variância podemos calcular o **desvio padrão** que é definido como a raiz quadrada da variância, ou ainda, a média da distância de cada valor até a média, que denota como a amostra é distribuída à medida em que afasta-se do valor médio.

Outro cálculo importante é o cálculo da **autocorrelação**, que é considerada uma medida de similaridade entre uma amostra e sua versão deslocada. A autocorrelação pode ser expressa na seguinte forma:

$$R_{xx}(l) = \sum_{n=i}^{N-|k|-1} x(n)x(n-l)$$

onde $x(n)$ é uma amostra de tamanho N. Este cálculo informa o quanto um determinado valor influencia os valores vizinhos, ou seja, o quão relacionadas estão os valores de uma amostra.

Ainda temos a **amplitude relativa**, que nada mais é do que a razão entre o valor mínimo e o valor máximo de uma determinada amostra, que demonstra características morfológicas da amostra.

3.2.2 Análise de Componentes Principais

A Análise de Componentes Principais é um método estatístico que efetua a redução da dimensão de matrizes originando uma matriz menos apenas com as informações principais da original. Nos métodos de classificação de batidas de eletrocardiograma serve para efetuar-se a redução do conjunto de características do sinal com perda mínima de informação. Por exemplo, considerando o conjunto de características a serem classificadas, é possível reduzir um conjunto de características de dimensão 10, transformando-o em um conjunto de dimensão 5 extraindo apenas as cinco maiores componentes principais.

O processo para encontrar componentes principais é o mesmo processo de encontrar autovetores de uma matriz de covariância R, formada pelo conjunto de características. A matriz de covariância R é dada por:

$$R = \frac{1}{N} \sum_{i=1}^{N} (F_i - \mu)(F_i - \mu)^T$$

Os autovalores e autovetores são obtidos resolvendo o problema do autovalor. Para este problema, os cinco autovetores, correspondentes aos cinco maiores autovalores, serão utilizados para a construção de um conjunto de características de dimensão reduzida.

Este novo conjunto de características é obtido pela multiplicação de cada vetor de características do conjunto original pelos cinco autovetores, y^k:

$$y^k = \Phi^T (F_1^{k*} - \mu)$$

3.2.3 Estatística de Ordem-Superior

Cumulantes são medidas de estatística de ordem superior comumente utilizados em técnicas de processamento de sinais e estatística. Eles podem ser vistos como um conjunto de valores que provém uma alternativa aos momentos de um sinal. Eles foram definidos de modo que duas distribuições probabilísticas semelhantes possuam cumulantes ainda mais semelhantes. O uso dos cumulantes na tarefa da classificação de ECGs ajuda a remover a variabilidade entre as batidas do mesmo tipo, e a amplificar as diferenças entre batidas de tipos diferentes [13]. Para a tarefa de classificação de batidas de ECG geralmente usam-se os cumulantes de primeira, segunda e terceira para auxiliar na formação do conjunto de características. Esses cumulantes são definidos por:

$$C_{2x}(k) = E\{x(n)x(n+k)\}$$

$$C_{3x}(k,l) = E\{x(n)x(n+k)x(n+l)\}$$

$$C_{4x}(k,l,m) =$$

$$E\{x(n)x(n+k)x(n+l)x(n+m)\} - C_{2x}(k)C_{2x}(m-l) - C_{2x}(l)C_{2x}(m-k) - C_{2x}(m)C_{2x}(l-k)$$

Onde E é o operador de esperança, que é definido como a soma das probabilidades do sinal multiplicada pelo valor da amostra. k, l e m são atrasos em tempo e o sinal x possui média zero (média das amostras é subtraída).

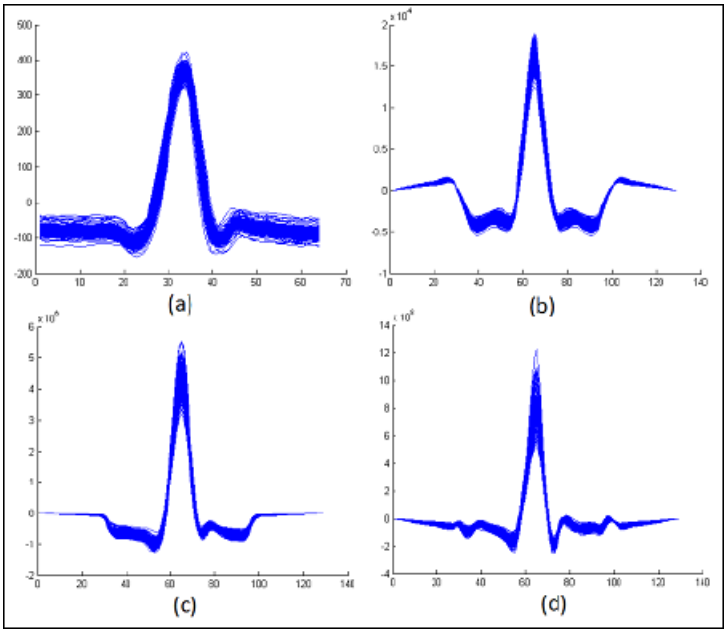

Figura 3.4: 200 amostras de batidas Normais: (a) sinal original, (b) cumulante de 2ª ordem, (c) cumulante de 3ª ordem e (d) cumulante de 4ª ordem

Como podem ser observados nas Figuras Figura 3.4, Figura 3.5 e Figura 3.6, os cumulantes amplificam as diferenças entre as batidas do tipo Normal (N), escape ventricular (PVC) e escape atrial (APB). Essas diferenças tornam-se ainda mais convincentes após a análise da diferença entre as magnitudes dos sinais originais e seus cumulantes. As batidas originais apresentam magnitudes semelhantes para todos os tipos. Entretanto, os cumulantes, além de apresentarem diferenças significativas para as magnitudes, essas diferenças não crescem de forma gradual ou seguindo qualquer tipo de padrão. Os valores máximos para os cumulantes de segunda ordem das batidas N, PVC e APB são, respectivamente, $2x10^4$, $8x10^4$ e $7x10^3$. Para os cumulantes de terceira ordem, as magnitudes são $6x10^6$, $1x10^7$ e $8x10^5$, e para os cumulantes de quarta ordem, $1.4x10^9$, $5x10^9$ e $8x10^7$. Demonstrando que a variação da magnitude das batidas em seus cumulantes pode ser útil na classificação dos ECGs.

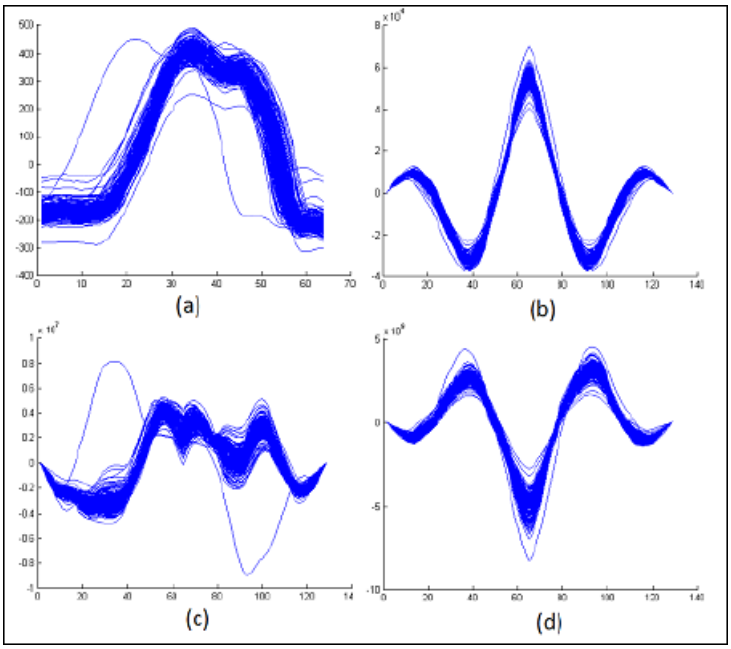

Figura 3.5: 200 amostras de batidas do tipo Escape Ventricular: (a) sinal original, (b) cumulante de 2ª ordem, (c) cumulante de 3ª ordem e (d) cumulante de 4ª ordem

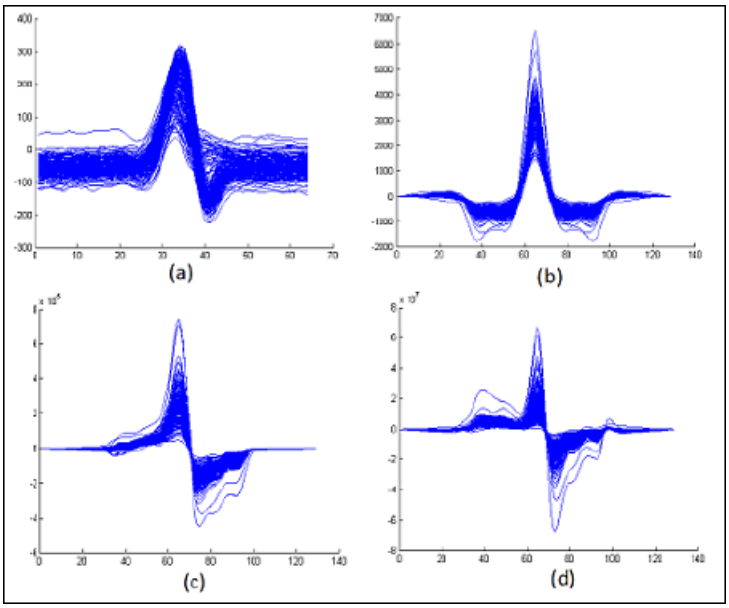

Figura 3.6: 200 amostras de batidas do tipo Escape Atrial: (a) sinal original, (b) cumulante de 2ª ordem, (c) cumulante de 3ª ordem e (d) cumulante de 4ª ordem

3.3 Redes Neurais Artificiais

Redes Neurais Artificiais são sistemas computacionais compostos por nós estruturados em uma rede, estes nós possuem ligações e seu funcionamento foi pensado em analogia ao comportamento de neurônios do cérebro. Atualmente as redes neurais são empregadas em inúmeras tarefas como aproximação de funções, reconhecimento de padrões, previsão de séries tempo-

rais e é claro, classificação. Existem inúmeros tipos de redes neurais, que diferem quanto ao algoritmo de aprendizado/treinamento de aprendizado, topologia de rede e comportamento dos neurônios.

3.3.1 Rede Neural Probabilística

A Rede Neural Probabilística (Probabilistic Neural Network - PNN) é uma rede neural utilizada basicamente como um classificador, ela é uma forma de implementação de um algoritmo de estatística chamado análise discriminante usando kernel. A rede é organizada em uma topologia composta por 4 camadas: Camada de Entrada, Camada de Padrão, Camada de Somatório e Camada de Saída. Topologia que pode ser visualizada na figura 3.7.

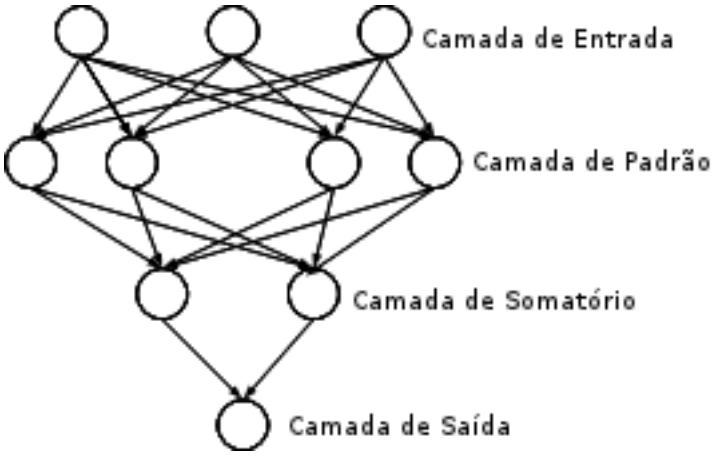

Figura 3.7: Topologia de uma PNN

A camada de entrada apenas distribui as amostras para os neurônios da camada de padrão, sendo que todas as entradas são propagadas para todos os neurônios da camada de padrão. Cada amostra do conjunto de treinamento corresponde à uma unidade da camada de padrão, e cada unidade de padrão estima a função de densidade de probabilidade (probability density function - PDF) da entrada ser parecida com a amostra do conjunto de treinamento na unidade e consequentemente ser classificada com a mesma classe da amostra do conjunto de treinamento. Cada classe que o classificador classifica corresponde a uma unidade de somatório que recebe a saída de todas as unidades de padrão cuja amostra de treinamento corresponde a classe referente à esta unidade. A probabilidade acumulada pela unidade de somatório é então transmitida para a camada de saída que decide a qual dentre as classes suportadas a amostra pertence.

Estimar a função densidade de probabilidade de uma determinada amostra é realizada na PNN utilizando a abordagem de estimação de densidade com kernel, também chamada de Janela de Parzen [14], que basicamente é uma técnica de interpolação de dados. Dada uma instancia

x, a janela de Parzen estima a PDF $\widehat{f_x}(x)$ para cada amostra do conjunto de dados. Conforme a seguinte expressão:

$$\widehat{f_x}(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x-x_i}{h}) \qquad (1)$$

onde K é a função kernel, h é um parâmetro de suavização e n é o número de amostras do conjunto de dados.

A função kernel mais utilizada é a Gaussiana que pode ser representada por:

$$H(v) = (2\pi)^{-\frac{1}{2}} e^{-\frac{1}{2}v^2}$$

Com isso, a expressão (1) é denotada por:

$$f(x) = \frac{1}{(2\pi)^{p/2}\sigma^p}\frac{1}{m}\sum_{i=1}^{m} exp[-\frac{(X-X_i)^t(X-X_i)}{2\sigma^2}]$$

Onde σ é o coeficiente de suavização, que determina o quanto as probabilidades são suavizadas.

Enfim, a PNN possui treinamento rápido, o conjunto de treinamento é utilizado diretamente no kernel, e converge para a superfície ótima de decisão Bayesiana.

3.3.2 Perceptron Multicamadas

Um Perceptron Multicamadas (MultiLayer Perceptron - MLP) é uma rede neural usualmente utilizada para resolver problemas de classificação. Um tipo comum de MLP é a rede neural Feed Forward Backpropagation (FFBNN), cujo nome demonstra que a entrada é propagada à frente na rede e os erros são retro propagados para corrigir os pesos dos neurônios. Para entender o comportamento da FFBNN, considere a topologia descrita na Figura 3.8.

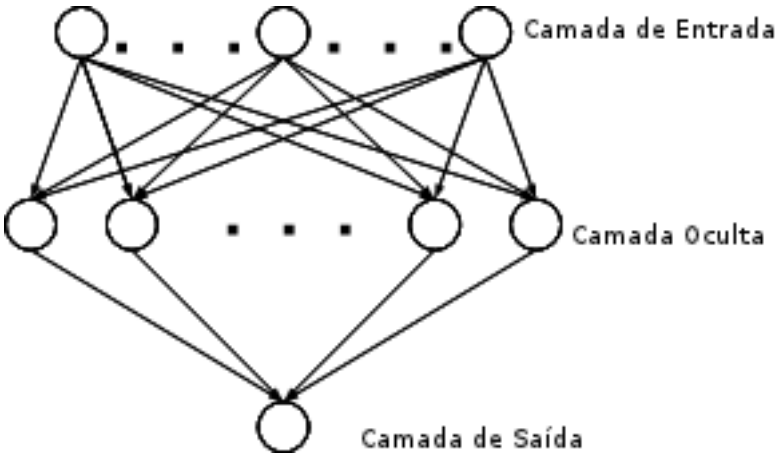

Figura 3.8: Topologia exemplo de uma FFBNN

O uso de redes neurais requer duas fases complementares: a fase de treinamento e a fase de teste. Durante a fase de treinamento, cada entrada do conjunto de treinamento é propagada através da camada de entrada da rede, camada que propaga as saídas de seus neurônios para os neurônios da camada oculta. Essa propagação se dá através da multiplicação do valor de entrada de cada neurônio da camada oculta com o respectivo peso do neurônio. A soma destes produtos por sua vez é à entrada de uma função de ativação que obtém um valor de ativação para cada neurônio da camada oculta. Os valores dos neurônios da camada de saída são calculados com o mesmo procedimento dos neurônios da camada oculta, entretanto, desta vez o valor final de ativação do neurônio é comparado como o valor desejado que está anotado junto ao conjunto de treinamento. A diferença entre o valor de ativação e o desejado é o erro para esta amostra que posteriormente é retro propagado para efetuar a correção dos pesos dos neurônios em todas as camadas. Esse processo é realizado durante um determinado número de épocas de treinamento.

Um método eficiente de retro propagação de erros, que minimiza o erro global da rede, foi proposto por Broyden-Fletcher- Goldfarb-Shanno (BFGS) como um algoritmo genérico de otimização. É um método não-linear quasi-Newton de otimização que transforma o problema do erro em um problema de minimização resolvido pela solução do problema de aproximação da matriz Hessiana [15].

A fase de testes apenas utiliza a rede previamente treinada, propagando as amostras do conjunto de teste e verificando se elas são classificadas corretamente.

3.4 Algoritmo de Vizinhança Mais Próxima

O algoritmo de k vizinhança mais próxima (k-nearest neighbor algorithm - KNN) é um algoritmo geralmente utilizado no reconhecimento de padrões, classificando uma determinada amostra de acordo com características de outras amostras vizinhas da amostra considerada. Uma das formas de determinar a proximidade das amostras é utilizando o cálculo da distância euclidiana de uma amostra de teste em relação a todas as amostras de treinamento. A batida de entrada é classificada utilizando a classe moda, ou seja, a classe mais frequente dentre as amostras mais próximas de acordo com a distância euclidiana [8].

A distância Euclidiana é definida como a distância entre dois pontos, sendo expressa como a raiz quadrada do quadrado da diferença entre os pontos:

$$d(x,y) = \sqrt{(x-y)^2}$$

A figura 3.9 apresenta um pseudo-código que demonstra o funcionamento do algoritmo

KNN. Primeiramente, são calculadas as distâncias da amostra a ser classificada com cada amostra do conjunto de treinamento. Essas distâncias são ordenadas e posteriormente se classifica a amostra com o tipo mais frequente dentre as k primeiras amostras, as amostras mais próximas, do vetor de distâncias ordenado.

```
Entrada: amostra, conjunto de treinamento, k
Saída  : tipo

1 for i ← 1 to #conjunto_treinamento do
2 |  distancias[i] ← distanciaEuclidiana(amostra, conjunto_treinamento[i]);
3 end
4 distancias ← Ordena(aux);
5 tipo ← moda(aux, k);
```

Figura 3.9: Pseudo código do algoritmo KNN

3.5 Resumo

Para realizar a tarefa de classificação de batidas de eletrocardiogramas algumas técnicas são necessárias para extrair características passíveis de classificação e ainda para a realização da classificação destas características.

Desta forma, para a realização da extração das características podem ser utilizadas a Transformada Discreta de Ondaleta, que transforma o sinal extraindo-se coeficientes de detalhe e de aproximação. Métodos estatísticos básicos como variância, autocorrelação e amplitude relativa. Análise de Componentes Principais que reduz a dimensão de conjuntos de características e ainda cumulantes (Estatísticas de Ordem Superior) que visam reduzir diferenças de características semelhantes.

Além disso, para a realização da classificação podem ser utilizadas Redes Neurais Probabilísticas, Algoritmos de Vizinhança Mais Próxima e ainda Redes Neurais Feed Forward Backpropagation.

4 *Métodos de Análise de Eletrocardiograma*

Este capítulo apresenta os principais métodos de classificação automática de ECG existentes na literatura, propõe algumas métricas quantitativas para a sua avaliação e finalmente realiza uma análise comparativa dos métodos descritos. Esta análise dá respaldo para a implementação dos métodos com maior acurácia, para que evidencie-se na prática o desempenho destes métodos.

Os métodos aplicam técnicas de diversas áreas da computação, bem como estatística computacional,reconhecimento de padrão, processamento de imagens, processamento de sinais, redes neurais artificiais, máquinas de vetores de suporte, etc. Entretanto existe um conjunto básico de passos comum entre os métodos: Detecção da batida, extração de características e classificação.

A detecção da batida é um processamento realizado sobre o sinal que consiste na detecção do complexo QRS, os métodos de detecção de QRS já possuem soluções excelentes e não serão abordados neste trabalho. O que é válido destacar é que após esta fase serão processados apenas segmentos de complexos QRS. A fase de extração de características consiste em transformar as informações do sinal em um formato adequado para a realização da classificação. Por último realiza-se a classificação das características extraídas em alguma classe dentre as classes que o método é capaz de identificar, como por exemplo batida normal, batida de marca-passo, etc.

Nas seções a seguir descreve-se a extração de características e classificação dos principais métodos existentes, estabelece-se métricas de avaliação quantitativa e por fim realiza-se uma análise comparativa dos métodos existentes.

4.1 Métodos Existentes

Nesta seção serão apresentados os principais métodos de classificação de batidas de eletrocardiograma existentes.

4.1.1 Método de Engin

A extração de características utilizada no método de Mehmet Engin [16] é baseada na Transformada Discreta de Ondaleta (Discrete Wavelet Transform - DWT) e em métodos estatísticos. São propostas três classes de características: cumulante de terceira ordem, modelos auto-regressivos e variância dos coeficientes de detalhe da transformada discreta de ondaleta em seis escalas diferentes. A função de ondaleta utilizada na DWT é a ondaleta de Daubechies de ordem 5 (db5).

O processo de classificação é realizado através de uma rede neural híbrida, que une conceitos de perceptron multi-camadas (MLP - Multilayer Perceptron), mapas auto- organizados (SOM - Self-Organized Map) e o algoritmo fuzzy c-means. O resultado desta combinação são camadas fuzzy auto-organizada conectada como uma MLP em cascata. O método classifica 4 tipos de batidas: Batida Normal, Batida com Onda P não-condutora, Batida com Extrassístole Atrial e Batida com Desvio de Ramo Direito. A acurácia obtida pelo método foi de 98%.

4.1.2 Método de Güler e Übeyli

No método de Inan Güler e Elif Derya Übeyli [17] é baseada na DWT e em métodos estatísticos, propõe-se quatro classes de características: média dos valores absolutos dos coeficientes de cada sub-banda, média da amplitude em cada sub-banda, desvio padrão dos coeficientes de cada sub-banda, razão entre a média absoluta de sub-bandas adjacentes. A função de ondaleta utilizada na DWT é a ondaleta de Daubechies de ordem 2 (db2).

A classificação é realizada por uma topologia combinada, usando 4 conjuntos de MLPs, uma para cada classe que o método classifica. O método propõe classificar 4 tipos de batidas: batida normal, batida com insuficiência cardíaca congestiva, taquicardia ventricular e fibrilação atrial. A acurácia obtida pelo método foi de 96,94%.

4.1.3 Método de Yu e Chen

Sung-Nien Yu e Ying-Hsiang Chen [18] criaram um método com extração de características baseada na DWT e em métodos estatísticos, são propostas quatro classes de características: variância dos coeficientes de cada um dos níveis, variância dos valores de autocorrelação dos coeficientes, a razão entre os valores máximos e mínimos das sub-bandas e a variância do sinal original. Essas 4 classes compõem o conjunto de características 1 (FS1), enquanto que o conjunto 2 (FS2) possui essas quatros classes mais o intervalo RR. A função de ondaleta utilizada na DWT é a ondaleta de Haar.

A classificação por sua vez é realizada através de uma Rede Neural Probabilística (Probabilistic Neural Network - PNN), com kernel Gaussiano. O método propõe classificar 6 tipos de batidas: batida normal, desvio de ramo direito, desvio de ramo esquerdo, extrassístole ventricular, extrassístole atrial e batida de marca-passo. A acurácia obtida pelo método foi de 99.32% com o conjunto FS1 e 99,65% com o conjunto FS2.

4.1.4 Método de Yu e Chou

Outro método existente foi criado por Sung-Nien Yu e Kuan-To Chou [19], a extração de características deste método é baseada na Análise de Componentes Independentes (Independent Component Analysis - ICA) e em métodos estatísticos, propõe-se dois conjuntos de componentes independentes, IC1 e IC2. O IC1 é obtido a partir do sinal referente a 1 segundo do ECG, sendo que o início é fixado a 0,722 segundos antes do ponto R, e o final em 0,278 segundos após o ponto R. Já para o IC2, 0,556 segundos são utilizados, sendo 0,278 segundos antes e 0,278 segundos após o ponto R. Em adição aos conjuntos de componentes independentes, o intervalo RR também é considerado uma característica, formando, analogamente, os conjuntos de características FV1 e FV2.

A classificação por sua vez é realizada por três algoritmos conhecidos na área da ACI: distância mínima Euclideana, distância mínima de Mahalonobis e um classificador Bayesiano baseado de erro mínimo. Além disso, foram alternados os classificadores responsáveis por cada um dos conjuntos de características. O método propõe classificar 6 tipos de batidas: batida normal, desvio de ramo direito, desvio de ramo esquerdo, extrassístole ventricular, extrassístole atrial e batida de marca-passo. A melhor acurácia obtida pelo método foi com o classificador Bayesiano para FV1 e a distância mínima Euclideana para FV2, alcançando uma acurácia de 99,51

4.1.5 Método de Chen e Yu

A extração de características, realizada no método de Ying-Hsiang Chen e Sung-Nien Yu [20], é baseada na DWT e em métodos estatísticos de alta ordem, propõe-se cinco classes de características: variância dos cumulantes de cada uma das sub-bandas, somatório normalizado dos cumulantes, o número total e ponto dos cruzamentos em zero dos cumulantes, os coeficientes de simetria dos cumulantes de terceira e quarta ordem, o intervalo RR atual, o intervalo RR anterior, e a razão entre os intervalos RR atual e anterior. A função de ondaleta utilizada na DWT é a ondaleta de Symlet de ordem 6 (sym6).

A classificação por sua vez é realizada por uma Rede Neural FFB (Feed-forward backpropagation Neural Network - FFBNN). O método propõe classificar 8 tipos de batidas: normal, bloqueio de ramo direito e esquerdo, extrassístole auricular e ventricular, batida de marca-passo, escape ventricular e flutter ventricular. Os autores organizaram o método em dois perfis de classificação, sendo o Perfil 1 composto por todos tipos de batidas, com exceção do escape ventricular e o flutter ventricular, e o Perfil 2 comporto por todos os tipos com exceção da batida de marca-passo. O Perfil 1 teve 99,70% de acurácia e o Perfil 2 apresentou uma acurácia de 97,53%.

4.1.6 Método de Minhas e Arif

O método de Fayyaz-ul-Amir Afsar Minhas e Muhammad Arif [21] tem uma extração de características baseada na DWT modificada e em métodos estatísticos, propõe-se cinco classes de características: variância dos coeficientes de cada um dos níveis, variância dos valores de autocorrelação dos coeficientes, a razão entre os valores máximos e mínimos das sub-bandas e a variância do sinal original e o intervalo RR. Adicionalmente, os autores utilizaram a PCA para reduzir a dimensionalidade do vetor para 5 características, o que permite que o tempo utilizado pela etapa da classificação seja reduzida. A função de ondaleta utilizada na DWT não é especificada.

A classificação por sua vez é realizada por um classificador do tipo KNN (k-nearest neighbor). O método propõe classificar 6 tipos de batidas: batida normal, desvio de ramo direito, desvio de ramo esquerdo, extrassístole ventricular, extrassístole atrial e batida de marca-passo. A acurácia obtida pelo método foi de 99,49% para o vetor de 11 características, utilizando o vetor reduzido, obtido com o uso da PCA, a acurácia foi 99,47%.

4.1.7 Método de Khadtare e Sahambi

A extração de características utilizada no método de Mahesh S. Khadtare e J.S. Sahambi [22] é baseada na DWT , propõe-se três classes de características: intervalos RR1, que é o intervalo entre a batida atual e a anterior, e o RR2, que é o intervalo entre a batida atual e a próxima batida do ECG. Adicionalmente, os autores utilizaram a PCA para reduzir a dimensionalidade do vetor para 5 características, o que permite que o tempo utilizado pela etapa da classificação seja reduzida. A função de ondaleta utilizada na DWT não é especificada.

A classificação por sua vez é realizada por DAGSVM (Dyrected Acyclic Graph SVM) na construção do modelo de classificação. Ela utiliza classificadores em uma topologia similar a uma árvore, onde os nodos folhas podem receber arestas de mais de um ramo. Quando uma instância entra no modelo, será direcionada a vários nodos de classificação, o qual resulta em positivo ou negativo para uma classe, de modo a alcançar algum nodo folha após algumas iterações. Ao final do processo, a classe do nodo folha com mais acessos será escolhida como a classe da instância. O método propõe classificar quatro tipos de batidas: batida normal, desvio de ramo direito, desvio de ramo esquerdo e extrassístole atrial. A acurácia obtida pelo método foi de 98,86%.

4.2 Métricas de Avaliação dos Métodos

Nessa seção apresentam-se as métricas que serão utilizadas para a avaliação dos métodos de classificação. Estas são baseadas em métricas de avaliação classificação de dados e de reconhecimento de padrões. São elas:

- Acurácia: mede a qualidade geral do método

$$acuracia = \frac{n^o\ de\ batidas\ classificadas\ corretamente}{n^o\ de\ batidas}$$

- Sensibilidade: mede a abrangência do método para um determinado tipo de batida anormal, ou seja, sua capacidade de detectar a presença dessa batida anormal sempre que ela existir

$$sensibilidade(x) = \frac{n^o\ de\ batidas\ anormais\ x\ classificadas\ corretamente}{n^o\ de\ batidas\ anormais\ x}$$

- Especificidade: mede a capacidade do método determinar que uma batida normal, ou

seja, sua capacidade de detectar a presença dessa batida normal sempre que ela existir

$$especificidade = \frac{n^o\ de\ batidas\ normais\ classificadas\ corretamente}{n^o\ de\ batidas\ normais}$$

Além destas duas métricas de desempenho, para a realização de uma comparação justa e posterior tomada de decisão de quais métodos são passíveis de uma implementação visando uma comparação de resultados, necessita-se que os métodos sejam implementados sobre o mesmo conjunto de batidas e classifiquem os mesmos tipos de batidas.

4.3 Métodos Propostos

Para a realização de uma comparação justa entre os métodos que evidencie o método com maior acurácia e com sensibilidades homogeneamente grandes, isto é, deseja-se um método que tenha uma taxa de classificações realizadas corretamente alta, medida pela acurácia do método e além disso taxas de classificação dado um tipo de batida igualmente alta, a sensibilidade. A sensibilidade assegura que todos os tipos de batidas de ECG foram propriamente classificados, evitando a superestimativa que a medida de acurácia pode causar. Além disso, novamente, é necessário que os métodos classifiquem os mesmos tipos de batidas e utilizem em seu treinamento e teste o mesmo número de batidas, provenientes preferencialmente do mesmo banco de dados. Esta seção apresenta uma análise comparativa entre os métodos descritos anteriormente e propõe a implementação de 3 destes métodos.

A tabela 4.1 apresenta dados retirados de 7 publicações de métodos de classificação de batidas de eletrocardiogramas existentes, na primeira coluna da esquerda para a direita temos o nome do método, na coluna central temos o número de tipos de batidas que o método classifica e na coluna à direita temos a medida de acurácia proposta pelos autores. A tabela está ordenada em ordem decrescente de acurácia. Nesta tabela evidencia-se que os métodos de Chen e Yu, Yu e Chen, Yu e Chou e Minhas e Arif são os métodos com maior acurácia. Além disso, estes métodos, exceto o Profile 2 do método de Chen e Yu, classificam os mesmos 6 tipos de batidas: Normal (N), Bloqueio de Ramo Esquerdo (LBBB), Bloqueio de Ramo Direito (RBBB), Extrassístole Ventricular (PVC), Extrassístole Atrial (APB) e Batida de Marca-passo (PB).

Com a análise da tabela 4.1 os métodos que seriam candidatos à implementação para posterior análise de resultados práticos são: Chen e Yu (Profile 1), Yu e Chen(FS1 e FS2), Yu e Chou e Minhas e Arif(com e sem PCA). Novamente, estes métodos possuem alta acurácia e classificam o mesmo conjunto de batidas.

A tabela 4.2 apresenta os valores de especificidade e sensibilidade relatados pelos autores

Tabela 4.1: Acurácia dos Métodos

Método	Tipos de batidas	Acurácia(%)
Chen e Yu (Profile 1)	6	99,70
Yu e Chen (FS2)	6	99,65
Yu e Chou	6	99,51
Minhas e Arif (sem PCA)	6	99,49
Minhas e Arif (com PCA)	6	99,47
Yu e Chen (FS1)	6	99.32
Khadtare e Sahambi	4	98,86
Engin	4	98,0
Chen e Yu (Profile 2)	7	97,53
Güler e Übeyli	4	96,94

Tabela 4.2: Análise dos Métodos

Método	Especificidade (%)	Sensitividade (%)				
		LBBB	RBBB	PVCV	APB	PB
Chen e Yu (Profile 1)	100	99,77	98,88	99,71	99,76	100
Yu e Chen (FS2)	99,97	99,33	99,54	99,04	99,76	100
Yu e Chou	99,66	99,59	99,25	99,57	98,24	100
Minhas e Arif (sem PCA)	99,84	99,21	99,38	98,93	99,23	99,97
Minhas e Arif (com PCA)	99,87	99,06	99,47	98,85	99,02	100
Yu e Chen (FS1)	99,86	99,04	99,42	97,65	98,82	100

dos métodos candidatos à implementação. Ela esta ordenada em ordem crescente de menor sensitividade de cada método. Percebe-se que, para os métodos considerados, os métodos com maior acurácia possuem também a sensitividades altas, o que de fato é desejado. Portanto continuamos com Chen e Yu (Profile 1), Yu e Chen(FS1 e FS2), Yu e Chou e Minhas e Arif(com e sem PCA) como os candidatos à implementação.

A tabela 4.3 apresenta a configuração dos conjuntos de treinamento e teste dos métodos propostos, em outras palavras, quantas batidas são utilizadas para a criação do conjunto de treinamento, que criará a estrutura necessária para a classificação das batidas, e para a criação do conjunto de testes, que será utilizado para verificar a acurácia da estrutura treinada. Neste momento evidencia-se que o Método de Chen e Yu utilizando o Profile 2, além de conforme anteriormente constatado classificar tipos de batidas diferentes, utiliza 7185 batidas em seu conjunto de treinamento e em seu conjunto de teste. O Método de Yu e Chou, mesmo classificando os mesmos tipos de batidas, acaba por utilizar apenas 6900 batidas em seu conjunto de treinamento e em seu conjunto de teste, enquanto que Chen e Yu (Profile 1), Yu e Chen(FS1 e FS2) e Minhas e Arif(com e sem PCA) utilizam a mesma configuração dos conjuntos de treinamento e de testes, 11600 batidas para cada.

Considerando esta análise, três métodos foram escolhidos para a implementação e real-

Tabela 4.3: Configuração dos conjuntos de treinamento e de teste

Método	# Conjunto de Treinamento	# Conjunto de Teste
Chen e Yu (Profile 1)	11600	11600
Yu e Chen (FS1 e FS2)	11600	11600
Minhas e Arif (com e sem PCA)	11600	11600
Chen e Yu (Profile 2)	7185	7185
Yu e Chou	6900	6900

ização desta análise comparativa de resultados: Método de Chen e Yu (Somente o Profile 1), Método de Minhas e Arif e Método de Yu e Chen. Essa escolha se baseou no fato de que esses três métodos são os mais eficientes de todos apresentados nas tabelas 4.1 e 4.2. Esses três métodos também classificam os mesmos seis tipos de batidas. Além disso, os três métodos utilizam o mesmo número de batidas provenientes do mesmo banco de dados para a validação de suas propostas, o banco de dados de arritmias do MIT-BIH que será melhor descrito no capítulo 5.

Soma-se aos métodos propostos, a implementação de um novo método de classificação de batidas de ECG, desenvolvido pelo mestrado de Fernando Arena Varella. Grande parte do aparato técnico utilizado pelo método foi desenvolvido pelo autor desse trabalho e a proposta do novo conjunto de características bem como do classificador faz parte do mestrado do Fernando.

4.4 Método de Varella e Lima

O método de Varella e Lima é baseado na transformada de ondaleta, Estatísticas de Ordem Superior (HOS) e no perceptron multicamadas (MultiLayer Perceptron - MLP). A extração características do sinal é realizada obtendo-se sub-bandas do sinal através de Transformada Discreta de Ondaleta (DWT). Posteriormente, um conjunto de características é construído utilizando HOS e medidas de estatística básica. Um MLP do tipo FFBNN classifica este conjunto de características entre seis tipos de arritmias em batidas de ECG. A acurácia obtida foi de 99,83% e as taxas de sensibilidade foram de no mínimo 99,48% para todos os tipos de batidas, o que supera os resultados dos outros métodos de classificação de batidas de ECG. O método de Varella e Lima utiliza o mesmo conjunto de gravações e classifica os mesmos tipos de batidas que os métodos selecionados.

4.5 Resumo

Os métodos de classificação de batidas de ECG possuem um conjunto básico de passos comum entre os métodos: Detecção da batida, extração de características e classificação.

Os principais métodos existentes são o Método de Engin, o Método de Güler e Übeyli, o Método de Chen e Yu, o Método de Yu e Chou, o Método de Yu e Chen, o Método de Minhas e Arif e o Método de Khadtare e Sahambi.

Para avaliá-los este trabalho verifica suas acurácias, sensitividades, bem como os tipos de arritmias que classificam e a configuração das suas amostras de teste.

Baseado na análise comparativa dos métodos existentes, este trabalho propõe a implementação do Método de Chen e Yu, do Método de Minhas e Arif e do Método de Yu e Chen. Juntamente com o Método de Varella e Lima proposto pelo mestrado de Fernando Arena Varela.

5 *Projeto da implementação*

Este capítulo descreve o projeto de implementação dos métodos de classificação de batidas de eletrocardiogramas de Chen e Yu, Minhas e Arif, Yu e Chen e Varella e Lima. Primeiramente será descrito o projeto da implementação da detecção e aquisição de batidas de eletrocardiogramas. Posteriormente será descrito o projeto da implementação das fases de extração de características e de classificação dos 4 métodos propostos.

5.1 Projeto da Detecção de Batidas

Nesta seção será apresentado o projeto da implementação da fase de detecção das batidas de eletrocardiogramas que serão utilizadas nos métodos de classificação.

O banco de dados de batidas de eletrocardiograma utilizado é o MIT-BIH Arrythmia Database que contém 48 trechos de meia hora de gravações de ECG de dois canais, obtidos de 47 sujeitos estudados pelo BIH entre os anos de 1975-1979. Vinte e três destas gravações foram escolhidas aleatoriamente de um conjunto de 4000 gravações de ECG de 24 horas de pacientes e ex-pacientes do Hospital Beth Israel de Boston(BIH). As 25 gravações restantes foram selecionadas do mesmo conjunto a fim de incluir as arritmias menos comuns porém significativas clinicamente que poderiam não ter sido incluídas na amostra aleatória. As amostras foram anotadas e/ou laudadas por no mínimo dois médicos.

O banco de dados contém para cada gravação 3 arquivos: Um arquivo com os sinais puros com a extensão dat, um arquivo com anotações de referência com a extensão atr e um arquivo com a descrição da gravação a extensão hea. O Arquivo de descrição da gravação contém inúmeras informações sobre a gravação, com a duração, número de amostras, taxa de amostragem, quantidade de sinais. Bem como informações sobre cada derivação presente na gravação, como o nome da derivação, ganho, valor inicial, etc. Na gravação de número 100, representada na tabela 5.1, tem-se que a duração do exame foi de 30 minutos 5 segundos e 556 milissegundos, com 650000 amostras, numa taxa de amostragem de 360 Hz, contendo dois sinais, cujas de-

scrições são separadas em duas partes, primeiramente é descrita a Derivação II e em seguida a Derivação V5. Na descrição de cada derivação está presente o arquivo de origem, a identificação da derivação, o ganho, o valor inicial e etc.

Tabela 5.1: Arquivo de descrição da gravação 100

```
Record 100
Notes
=====
=====
Starting time: not specified
Length: 30:05.556 (650000 sample intervals)
Sampling frequency: 360 Hz
2 signals
Group 0, Signal 0:
   File: 100.dat
   Description: MLII
   Gain: 200 adu/mV
   Initial value: 995
   Storage format: 212
   I/O: can be unbuffered
   ADC resolution: 11 bits
   ADC zero: 1024
   Baseline: 1024
   Checksum: -22131
Group 0, Signal 1:
   File: 100.dat
   Description: V5
   Gain: 200 adu/mV
   Initial value: 1011
   Storage format: 212
   I/O: can be unbuffered
   ADC resolution: 11 bits
   ADC zero: 1024
   Baseline: 1024
   Checksum: 20052
```

Os arquivos com a extensão dat contém valores para duas derivações de ECG, cada linha contém uma amostra de ambas as derivações presentes nesta gravação separadas por ⁘ Por exemplo, a gravação de número 100, contém duas derivações II e V5, as 10 primeiras amostras de cada derivação são:

995;1011;
995;1011;
995;1011;

995;1011;
995;1011;
995;1011;
995;1011;
995;1011;
1000;1008;
997;1008;

Vale ressaltar que os arquivos de sinais (.dat) disponíveis no site do banco de dados, possuem uma codificação própria, devido à este fato é necessário utilizar uma função de decodificação escrita na linguagem C. Esta função está disponível em [23] e monta arquivos de sinais da forma especificada anteriormente.

O arquivo de anotações contém três informações importantes: o tempo em que o pico da onda R ocorre, o número da amostra que este piuco ocorre e a informação do tipo da batida que contém este pico. As 10 primeiras linhas das anotações da gravação de número 100 são:

0:00.050	18	+	0	0	0	(N
0:00.214	77	N	0	0	0	
0:01.028	370	N	0	0	0	
0:01.839	662	N	0	0	0	
0:02.628	946	N	0	0	0	
0:03.419	1231	N	0	0	0	
0:04.208	1515	N	0	0	0	
0:05.025	1809	N	0	0	0	
0:05.678	2044	A	0	0	0	
0:06.672	2402	N	0	0	0	

Na primeira linha tem-se que aos 50 ms, na amostra 18 houve uma batida anotada como + e (N. Isso significa que esta batida é a primeira do exame e ela é Normal (N). Entretanto na nona linha tem-se que aos 5,678 segundos, na amostra 2044 uma batida é anotada como A, que corresponde a uma Extrassístole Atrial. Os símbolos importantes para este trabalho são N para Batida Normal, L para Bloqueio de Ramo Esquerdo,R para Bloqueio de Ramo Direito, A para Extrassístole Atrial, V para Extrassístole Ventricular e / para Batida de Marca-passo.

Para processar estes dados de forma simples e organizada, pensou-se em implementar 4 classes: Classe Record, Classe Lead, Classe Beat e Classe BeatType. Conforme pode ser visualizado na figura 5.1. O projeto da classe Beat prevê a implementação de cinco atributos:

- uma lista de amostras (valores) que contém uma batida de ECG, sampleList.

- o tempo no qual a batida ocorre, time.
- o tipo que a batida foi anotada, type.
- o intervalo RR, ou seja, a distância entre a batida atual e a antecessora, RRinterval.
- o intervalo RR da batida anterior, previousRRInterval.

Além de operações básicas de acesso (get) e armazenamento dos atributos (set), tem-se a operação de armazenar uma amostra na lista de amostras (setSample). A classe depende da classe BeatType, que nada mais é uma estrutura para organizar os tipos de batidas de ECG utilizados pelos métodos.

A classe Lead depende da classe Beat e prevê a implementação de três atributos:

- uma lista de batidas (lista de amostras) que contém uma derivação de ECG (beatList).
- o nome da derivação (name).
- a linha de base da derivação (baseline).

Novamente, além de operações básicas de acesso (get) e armazenamento dos atributos (set), tem-se a operação de armazenar uma batida na lista de batidas (setBeatIntervalToLead).

Finalmente, a classe Record depende da classe Lead e estrutura apenas dois atributos:

- uma lista de derivações (leadsList).
- um identificador para a gravação (id).

Mais uma vez, além de operações básicas de acesso (get) e armazenamento dos atributos (set), tem-se a operação criação de uma gravação (createRecord) que a partir de um conjunto de sinais, um arquivo de anotações e um arquivo de descrição, estrutura os dados de uma gravação de ECG.

A estruturação dos dados provenientes dos arquivos do banco de dados utilizado é considerada a fase de Detecção de Batidas e é comum para os quatro métodos propostos neste trabalho.

5.2 Projeto da Extração de Características

Nesta seção será apresentado o projeto de implementação das fases de extração de características dos quatro métodos propostos por este trabalho. Devido ao fato que os quatro métodos

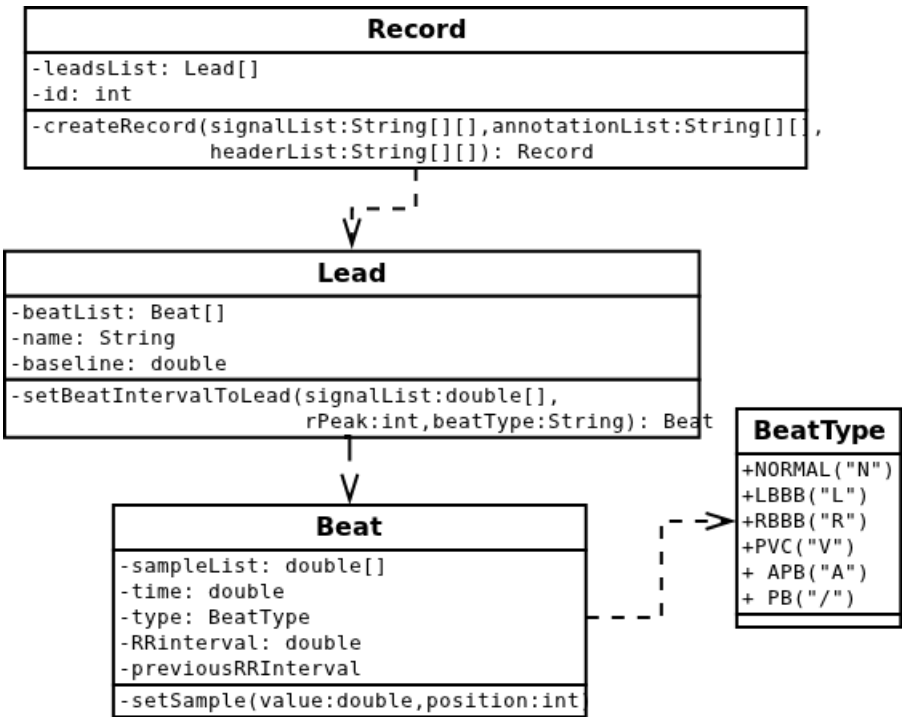

Figura 5.1: Classes Record, Lead, Beat e BeatType

utilizam a Transformada Discreta de Ondaleta na etapa inicial da fase de Extração de Características, uma única classe DWT implementará a DWT para os quatro métodos. Ainda tem-se que estes métodos utilizam estatísticas básicas para extrair as características, que são projetadas como uma classe chamada MathSupport. O diagrama de classe dessas classes está presente na figura 5.2.

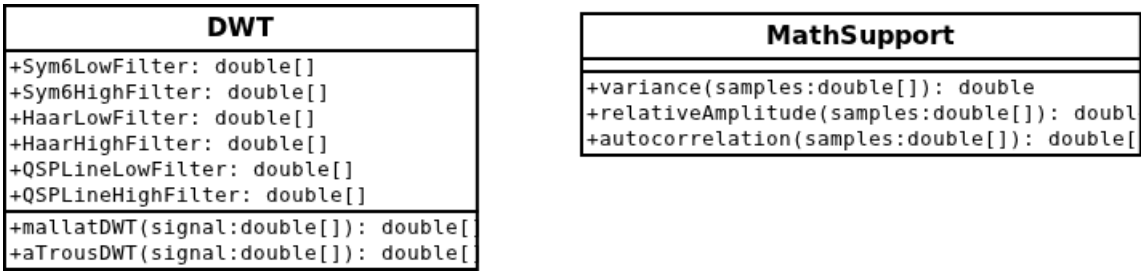

Figura 5.2: Classes DWT e MathSupport

O projeto da classe DWT prevê 6 atributos para os filtros de passa-baixa e passa-alta de três tipos de wavelets, a Symlet 6, a ondaleta de Haar e a Spline Quadrática. As operações previstas são a aplicação do algoritmo piramidal de Mallat e o algoritmo à trous, descritos anteriormente no capítulo 3. Já a classe MathSupport apenas apresenta as três medidas estatísticas comuns aos métodos, a variância, amplitude relativa e autocorrelação.

5.2.1 Método de Chen e Yu

Recapitulando, durante a sua fase de extração de características o método proposto por Chen e Yu utiliza a Transformada Discreta de Ondaleta, juntamente com cumulantes e estatística básica para extrair um grande conjunto composto por 30 características de cada batida de ECG.

Primeiramente o método da Classe DWT à trous, com a ondaleta mãe symlet 6, para calcular cinco níveis de coeficientes. Posteriormente, as três maiores cumulantes da HoS são utilizadas para ajudar a construir o conjunto de características, nominalmente: cumulante de segunda ordem, cumulante de terceira ordem e cumulante de quarta ordem. As cumulantes de segunda, terceira e quartas ordem foram extraídas dos coeficientes de detalhe de 3º (D3), 4º (D4) e 5º (D5) nível, totalizando nove cumulantes são calculadas para cada batida. Com isso, algumas as seguintes características são calculadas:

- Variância das cumulantes (9 características).
- Somatório normalizado para cada cumulante (9 características).
- Número de cruzamentos à linha de base zero das cumulantes provenientes dos coeficientes de detalhe de 5º nível (3 características).
- A simetria das cumulantes de terceira e quarta ordem (6 características).
- intervalo RR atual (uma característica).
- intervalo RR anterior (uma característica).
- a razão entre o intervalo RR atual e o anterior (uma característica).

Somando os valores entre parenteses apresentados na enumeração anterior, temos as 30 características propostas pelo método.

O projeto da fase de extração de características para o Método de Chen e Yu, necessita, além das classes DWT e MathSupport descritas anteriormente, de outras duas classes, a Classe ChenYuFeatureExtractor e a Classe Cumulants. Todas as classes utilizadas na extração de características do método são apresentadas no diagrama de classe da figura 5.3.

Além das já descritas classes DWT e MathSupport, tem-se mais duas classes, a Classe Cumulants que implementa as operações de cálculo de cumulante de segunda ordem (cumulant2x) terceira ordem (cumulant3x) e quarta ordem (cumulant4x). Enquanto que a classe ChenYuFeatureExtractor contém dois atributos (trainingSet e testingSet) que nada mais são do que uma

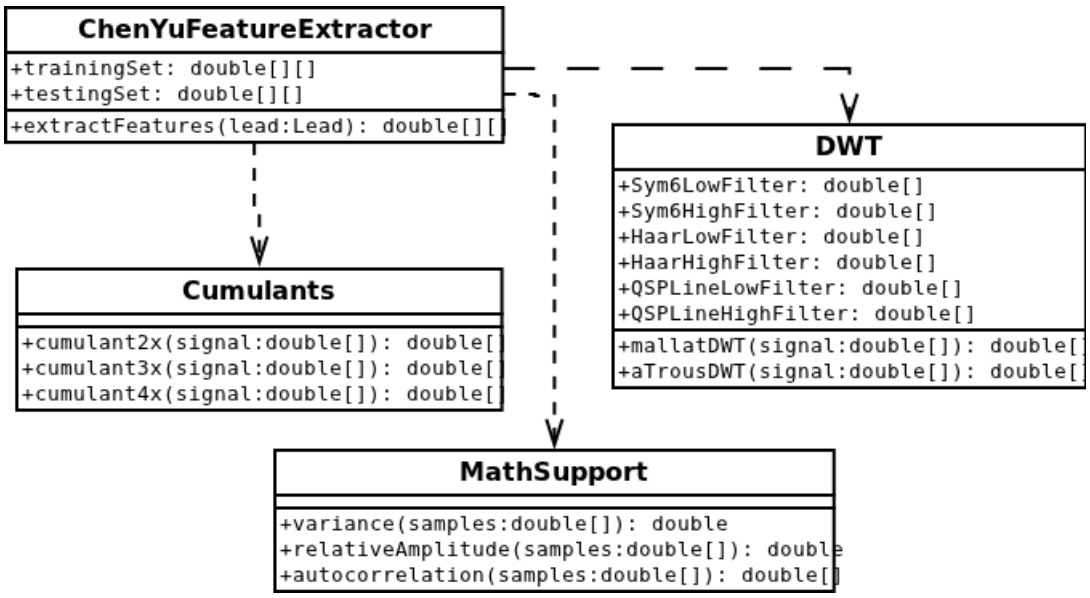

Figura 5.3: Classes para o Método de Chen e Yu

matrizes bxf, de b batidas com f características. Além da operação de extrair as características de uma determinada Lead, chamada de extractFeatures que, com o suporte das classes DWT, Cumulants e MathSupport, calcula as 30 características para cada batida da derivação (lead).

5.2.2 Método de Yu e Chen

O método proposto por Yu e Chen é baseado no uso da Transformada Discreta de Ondaleta e métodos estatísticos para a Extração de Características das batidas de ECG que são classificadas com uma Rede Neural Probabilística (Probabilistic Neural Network - PNN).

A Extração de Características é realizada primeiramente pela aplicação da DWT. A DWT proposta utiliza o algoritmo piramidal proposto por Mallat onde uma série de filtros passa-baixas e passa-altas são aplicados. A partir das sub-bandas D1, D2 e A2 são calculadas as seguintes características:

- a variância do sinal e dos coeficientes (4 características).
- a variância da auto-correlação das sub-bandas (3 características).
- a amplitude relativa das sub-bandas (3 características).
- o intervalo RR atual (uma característica).

São construídos assim dois conjuntos de características, nominalmente, FS1 e FS2. O FS1 inclui a variância do sinal original, a variância de D1, D2 e A2, a variância da auto-correlação

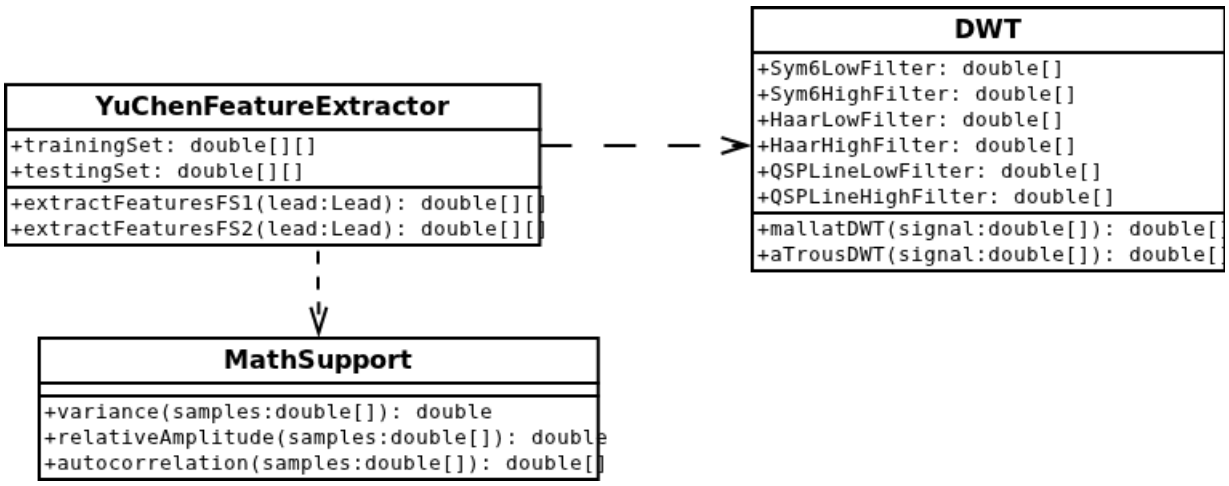

Figura 5.4: Classes para o Método de Yu e Chen

de D1, D2 e A2 e a amplitude relativa de D1, D2 e A2, totalizando 10 características. O FS2 contém as 10 características do FS1 e o intervalo RR atual que informa a distância entre duas batidas consecutivas, dado que esta medida é muito importante para detecção de arritmias. Após a extração de características, elas são normalizadas usando a função sigmóide tangente hiperbólica para mapear os valores de cada conjunto de características no intervalo [-1,+1].

Desta vez o projeto da fase de extração de características fica mais simples, pois necessita-se apenas do projeto de mais uma classe, a classe YuChenFeatureExtractor, além é claro das classe DWT e MathSupport já descritas. Com isso o diagrama de classe para o projeto se dá como representado na figura 5.4.

A classe YuChenFeatureExtractor contém quatro atributos, trainingSet e testingSet de FS1 e FS2, que nada mais são do que uma matrizes bxf, de b batidas com f características. Além das operações de extração das características de uma determinada Lead, chamada de extractFeaturesFS1 que, com o suporte das classes DWT e MathSupport, calcula as características para cada batida da derivação (lead).

5.2.3 Método de Minhas e Arif

O método desenvolvido por Minhas e Arif é similar ao de Yu e Chen. Eles usaram as mesmas medidas estatísticas para construir o conjunto de características, mas existem algumas diferenças na DWT.

Primeiramente aplica-se uma re-amostragem nas gravações de 360 Hz para 250 Hz. posteriormente é aplicado o algoritmo DWT à trous com o uso da ondaleta Spline Quadrática como ondaleta mãe. Os coeficientes de detalhe de primeiro e segundo nível, bem como os coeficientes

de aproximação do segundo nível, foram selecionados como uma base para o conjunto de características. Para eles, as mesmas características estatísticas como o FS2 do método de Yu e Chen são calculadas. São elas:

- a variância do sinal e dos coeficientes (4 características).
- a variância da auto-correlação das sub-bandas (3 características).
- a amplitude relativa das sub-bandas (3 características).
- o intervalo RR atual (uma característica).

Outro conjunto de características foi implementado empregando Análise de Componentes Principais para gerar um segundo conjunto de características, com dimensão reduzida. Considerando o conjunto de características anteriormente proposto, as 10 primeiras características apresentadas são transformadas, em um conjunto de dimensão 5 características, extraindo apenas as cinco maiores componentes principais do conjunto.

A figura 5.5 apresenta o diagrama de classe para o projeto de implementação da extração de características para este método. Nele, além das classes DWT e MathSupport, tem-se mais duas classes a classe MinhasArifFeatureExtractor e a classe PCA.

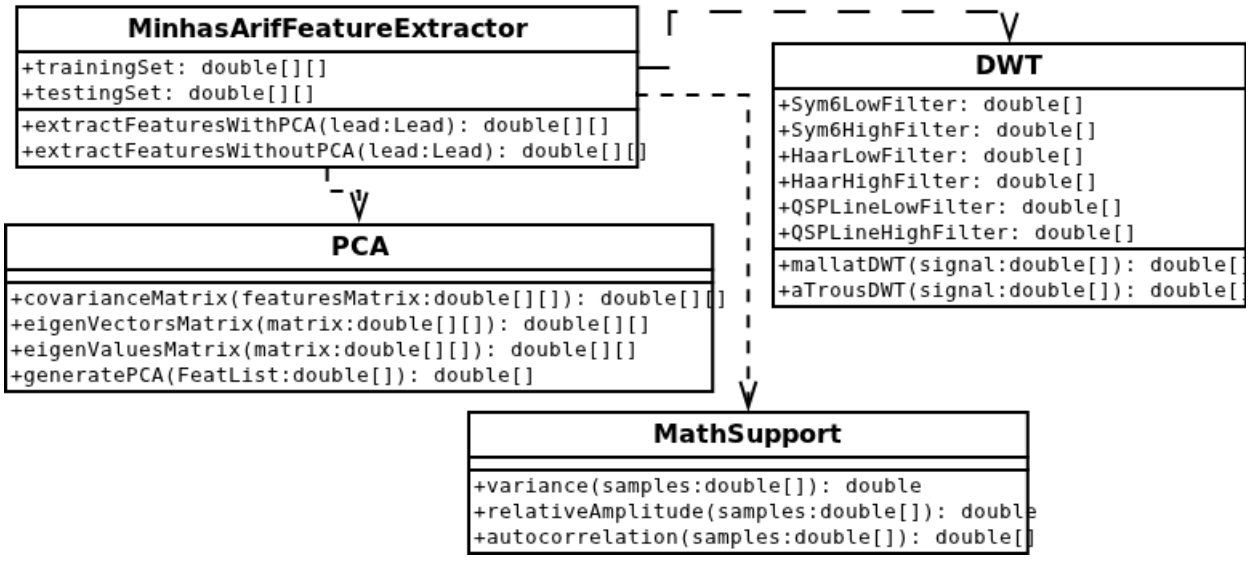

Figura 5.5: Classes para o Método de Minhas e Arif

A classe PCA implementa a analise de componentes principais, analise que depende das suas operações de criação da matriz de covariância, covarianceMatrix, criada a partir da matriz de características extraídas, a operação de obtenção de autovetores (eigenVectorsMatrix) e autovalores (eigenValuesMatrix) da matriz de covariância e por fim a operação de geração dos

componentes principais (generatePCA). Esta classe juntamente com a DWT e MathSupport são utilizadas pela classe MinhasArifFeatureExtractor para a geração de conjuntos de características com PCA, pela operação extractFeaturesWithPCA, e sem PCA , pela operação extractFeaturesWithoutPCA.

5.2.4 Método de Varella e Lima

O método proposto utiliza a transformada de ondaleta à trous, com a Spline Quadrática como ondaleta-mãe por ela ter mostrado, no método de Minhas e Arif, melhorias na classificação de batidas em relação à ondaleta de Haar, e por possuir filtros de tamanhos reduzidos em relação à Sym6, o que ajuda a reduzir o tempo necessário para a aplicação da DWT. A Spline Quadrática é definida pelos seguintes filtros passa-altas e passa-baixas: h=[2.0,-2.0] e l=[0.125, 0.375, 0.375, 0.125].

Além do uso de DWT, este método,assim como o método de Yu e Chen, utiliza Cumulantes. O seu uso na tarefa da classificação de ECGs ajuda a remover a variabilidade entre as batidas do mesmo tipo, e a amplificar as diferenças entre batidas de tipos diferentes. No método proposto utilizados os cumulantes de segunda, terceira e quarta ordem para auxiliar na formação do conjunto de características.

Para construir o conjunto de características deste método foi selecionado um total de onze características extraídas dos coeficientes de ondaleta e de seus cumulantes. Oito características são idênticas às utilizadas no método proposto por Yu e Chen, enquanto que três características foram selecionadas a partir da análise da variabilidade dos cumulantes dos coeficientes de ondaleta. A definição do conjunto de características irá considerar os seguintes sinais em sua definição: o sinal original da batida (S), seus coeficientes de detalhe da primeira e segunda sub-bandas (D1 e D2) e de aproximação da segunda sub-banda (A1), obtidos pela aplicação da DWT, o cumulante de segunda ordem de D1 (C2D1) e os cumulantes de terceira e quarta ordem de D2 (C3D2 e C4D2). A partir dos sinais definidos acima, um conjunto de medidas tais como a autocorrelação, variância e amplitude relativa são obtidas para que o conjunto de características seja formado.

Considerando então os sinais e as medidas estatísticas descritas acima, o conjunto de características utilizadas por este método é constituído por:

- a variância de S, D1, D2 e A2. (4 características)
- a variância da autocorrelação de D1, D2 e A2. (3 características)

- a amplitude relativa de C2D1 e C4D2. (2 características)
- a variância da autocorrelação de C2D1. (1 característica)
- o intervalo entre as ondas R da batida atual e a batida anterior (intervalo RR). (1 característica)

Totalizando 11 características.

O projeto da fase de extração de características deste método é semelhante ao descrito para o método de Chen e Yu, necessitando além das classes DWT, MathSupport e Cumulants uma classe extra, a Classe VarellaLimaFeatureExtractor que também contém dois atributos, trainingSet e testingSet. Além da operação de extrair as características de uma determinada Lead, chamada de extractFeatures que, com o suporte das classes DWT, Cumulants e MathSupport, calcula as 11 características para cada batida da derivação (lead).

5.3 Projeto da Classificação

Esta seção apresenta o projeto de implementação da fase de classificação dos quatro métodos propostos por este trabalho.

5.3.1 Método de Chen e Yu

A fase de classificação desse método é realizada por uma MLP do tipo FFBNN composta neste caso por três camadas de neurônios. A rede neural Feed Forward Backpropagation (FFBNN), cujo nome demonstra que a entrada é propagada à frente na rede e os erros são retro propagados para corrigir os pesos dos neurônios. O método de Broyden-Fletcher-Goldfarb-Shanno (BFGS) é utilizado para a retropropagação dos erros e consequente correção dos pesos dos neurônios. O uso de redes neurais requer duas fases complementares: a fase de treinamento e a fase de teste. A topologia de rede é apresentada na figura 5.6, ela apresenta 30 neurônios de entrada, $i_1..i_{30}$,(um para cada característica), 60 neurônios na camada oculta, $h_1...h_{60}$, e apenas um neurônio na camada de saída, o_1.

O projeto de implementação da fase de classificação deste método é apresentado com o auxílio da figura 5.7 que contém as três classes responsáveis pela implementação. A classe FFBNN que implementa uma rede neural do tipo FFBNN, cujos atributos são informações sobre a topologia da rede, número de neurônios da camada de entrada, oculta e de saída, potenciais de ativação dos neurônios, o erro delta na camada de saída e na camada oculta, a matriz com

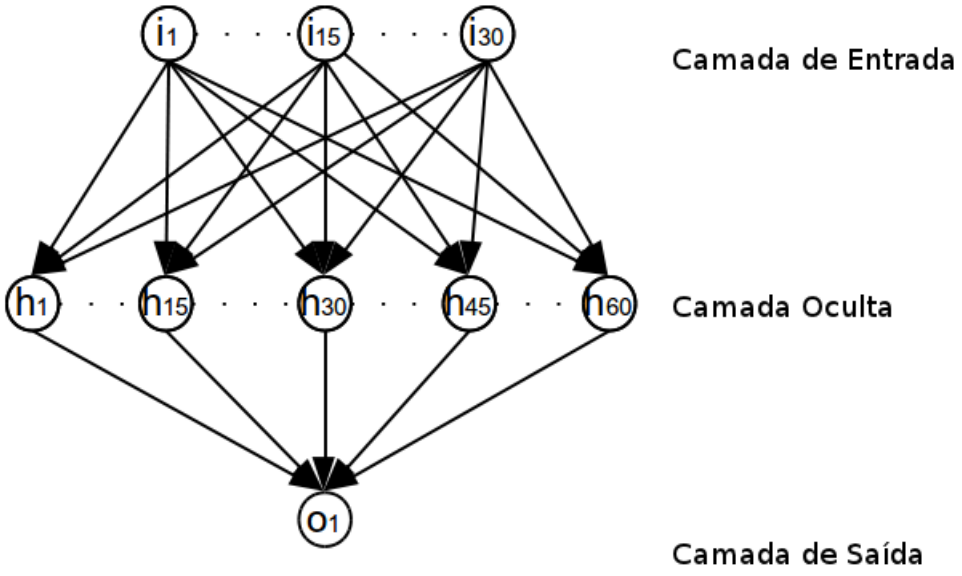

Figura 5.6: Topologia da FFBNN de Chen e Yu

os pesos dos neurônios ocultos (por exemplo.: de tamanho 30 neurônios de entrada por 60 neurônios ocultos) e de saída (60x1). Além da cópia de ambas as matrizes na época de treinamento anterior. Esta classe contém duas operações principais, a função de treinamento (train) e de teste (test). A operação operação de treinamento, treina a estrutura baseando-se nas amostras (patterns) provenientes do conjunto de treinamento com suas respectivas classes (targets), uma quantidade de épocas de treinamento (epochs), um coeficiente de aprendizado (learningRate) que determina a proporção com que os pesos são modificados, um momentum que determina quanto dos pesos anteriores é conservado e um valor mínimo de erro que a rede neural admite (minError).

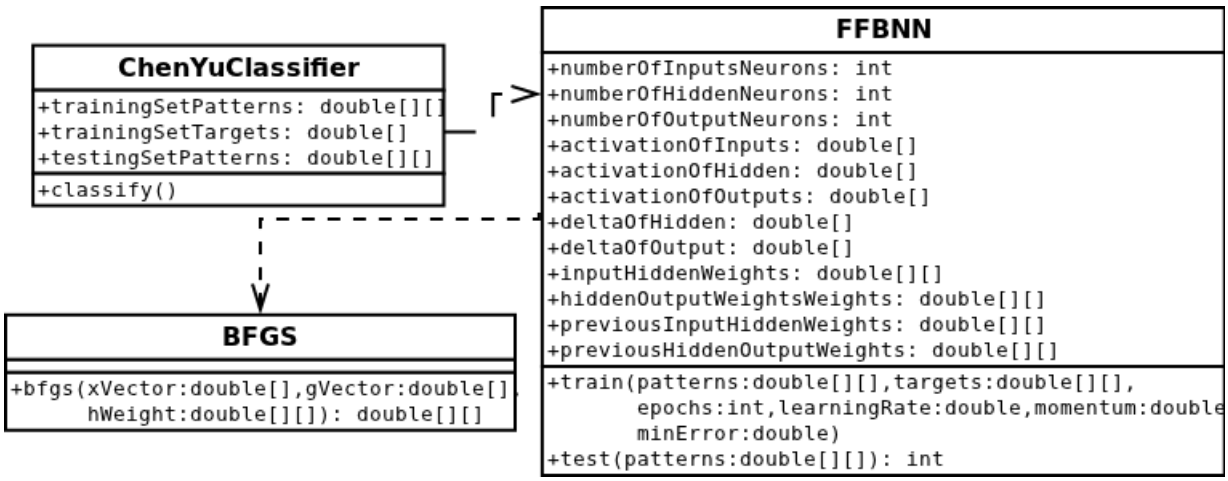

Figura 5.7: Classes para o Método de Chen e Yu

A classe ChenYuClassifier armazena o conjunto de treinamento(trainingSetPatterns) e as

classes do próprio (trainingSetTargets), bem como o conjunto de teste (testingSetPatterns). Além disso, contém a operação classify, que treina e testa a rede neural para os conjuntos de características do método.

5.3.2 Método de Yu e Chen

Yu e Chen utilizam uma PNN para realizar a classificação das batidas. Relembrando, a PNN é uma rede neural utilizada basicamente como um classificador, sendo uma forma de implementação de um algoritmo de estatística chamado análise discriminante usando kernel.

A rede é organizada em uma topologia composta por 4 camadas: Camada de Entrada, Camada de Padrão, Camada de Somatório e Camada de Saída. A topologia deste método é apresentada na figura 5.8 nela uma amostra com 10 (FS1) ou 11 (FS2) características é propagada pela amada de entrada para cada um dos 11600 neurônios da camada de padrão (um para cada amostra do conjunto de treinamento), a saída da camada de padrão é propagada à camada de somatório da classe cujo neurônio de padrão pertence, com isso temos 6 unidades de somatório que somam as probabilidades da amostra pertencer à classe, a saída da camada de somatória é propagada à camada de saída que verifica qual é a classe mais provável para a amostra.

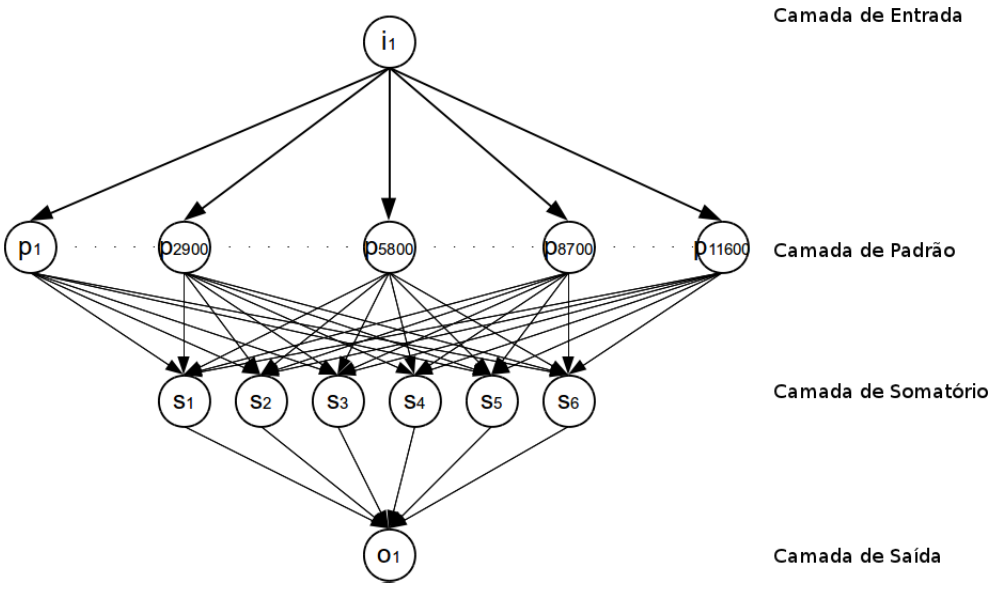

Figura 5.8: Topologia da PNN de Yu e Chen

A figura 5.9 apresenta o projeto da implementação da fase de classificação para este método, contendo o diagrama de classes. Duas classes se fazem necessárias, a classe que implementa a rede PNN e a classe que organiza o classificador do método (YuChenClassifier). A classe PNN implementa uma rede neural probabilística, seus atributos são o número de características da amostra a ser classificada (numberOfFeatures), o tamanho do conjunto de treinamento que

determinará a quantidade de neurônios de padrão (trainingSetSize) e a quantidade de elementos de cada uma das 6 classes (classes[6]). A operação gaussianPropagate, propaga uma amostra (sample) do conjunto de testes, com um determinado coeficiente de suavização (sigma) em uma rede com um determinado conjunto de treinamento (trainingSet).

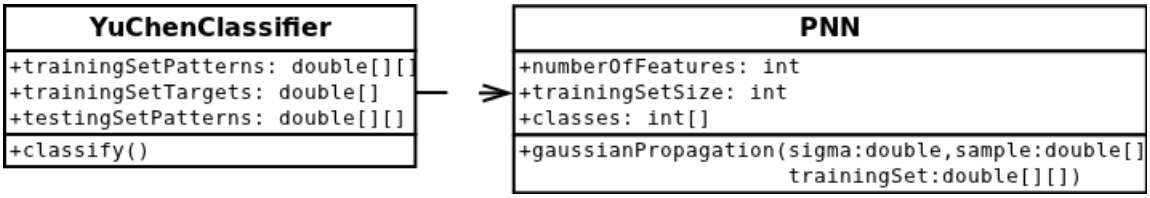

Figura 5.9: Classes para o Método de Yu e Chen

Esta classe é usada pela classe YuChenClassifier que armazena um conjunto de treinamento, um conjunto de testes e as classes das amostaras do conjunto de treinamento e classifica utilizando a operação classify que por sua vez organiza chamadas para a função gaussianPropagate da classe PNN.

5.3.3 Método de Minhas e Arif

Minhas e Arif utilizaram o algoritmo de k vizinhança mais próxima para classificar p conjuntos de testes. O algoritmo de k-vizinhança mais próxima calcula a distância euclidiana de uma amostra de teste em relação a todas as amostras de treinamento. A batida de entrada é classificada utilizando a classe moda, ou seja, a classe mais frequente dentre as amostras mais próximas de acordo com a distância euclidiana.

O projeto de implementação da fase de classificação deste método conta com duas classes, KNN e MinhasArifClassifier, que são apresentadas pelo diagrama de classes da figura 5.10.

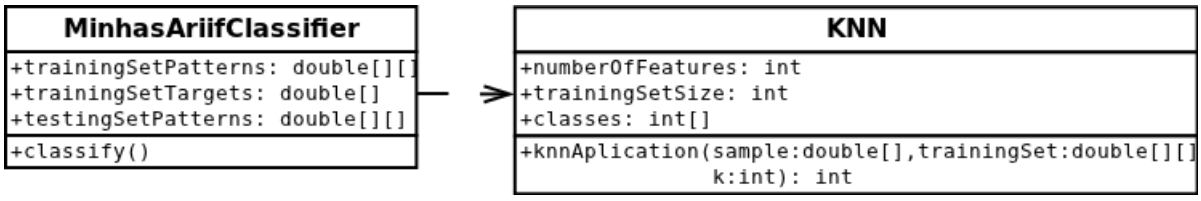

Figura 5.10: Classes para o Método de Minhas e Arif

A classe MinhasArifClassifier que armazena um conjunto de treinamento, um conjunto de testes e as classes das amostaras do conjunto de treinamento e classifica utilizando a operação classify, que por sua vez utiliza a classe KNN que por sua vez utiliza a classe KNN, cujos atributos são o número de características da amostra a ser classificada (numberOfFeatures), o tamanho do conjunto de treinamento que determinará a quantidade de amostras para qual a

distância euclidiana será calculada(trainingSetSize) a quantidade de elementos de cada uma das 6 classes (classes[6]). A classe implementa o algoritmo KNN que dada uma amostra do conjunto de testes, calcula a distância com todas as amostras do conjunto de treinamento, e verifica qual a classe que a amostra mais se aproxima.

5.3.4 Método de Varella e Lima

A classificação neste método é realizada também por uma uma rede neural FFBNN cuja topologia foi definida empiricamente, através do teste extensivo de diversas topologias alternativas, é apresentada na Figura 5.11.

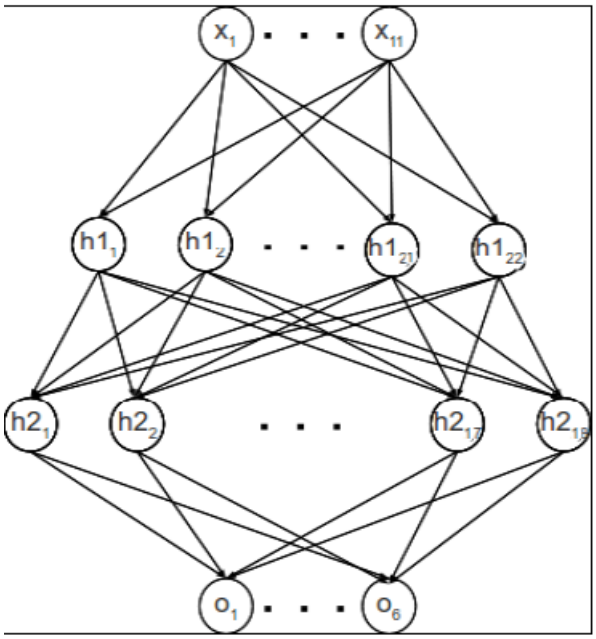

Figura 5.11: Topologia da FFBNN do Método de Varella e Lima

Como pode ser observado na Figura 5.11, a rede neural utilizada é composta por quatro camadas: a camada de entrada (x), com onze neurônios, as camadas ocultas (h1) e (h2), com vinte e dois e dezoito neurônios, respectivamente, e a camada de saída, com seis neurônios. Os valores que servem como entrada na primeira camada são as características da batida, por isso, essa camada deve possuir tantos neurônios quanto seja o tamanho do conjunto de características. Neste método, a função de ativação escolhida é a tangente hiperbólica sigmóide (tansig). O método de Broyden-Fletcher-Goldfarb-Shanno (BFGS) é utilizado para efetuar a retro-propagação do erro e minimizar o erro geral da rede. que foi originalmente

Como a diferença do classificador deste método com o método de Chen e Yu é apenas a topologia da rede, o projeto deste classificador é similar ao projeto de Chen e Yu, com a

exceção de que se projetou uma classe VarellaLimaClassifier, que também armazena o conjunto de treinamento(trainingSetPatterns) e as classes do próprio (trainingSetTargets), bem como o conjunto de teste (testingSetPatterns). Além disso, contém a operação classify, que treina e testa a rede neural para os conjuntos de características do método.

5.4 Resumo

O Projeto da Fase de Detecção de Batidas é comum para todos os métodos, as batidas provém do banco de dados MIT-BIH Arrythmia Database. Neste projeto são previstas quatro classes para estruturar as informações do banco de dados, a Classe Record, que estrutura uma gravação, a classe Lead que estrutura uma derivação, a classe Beat que estrutura uma batida de ECG e a classe BeatType que estrutura os tipos de batidas.

O Projeto da Extração de Características prevê duas classes comuns à todos os métodos, a Classe DWT, que implementa a Transformada Discreta de Ondaleta e a classe MathSupport com funções comuns de estatística. A classes PCA, que faz a Análise de Componentes Principais e a Classe Cumulants que implementa as cumulantes são utilizadas por alguns métodos. Além das classes ChenYuFeatureExtractor, YuChenFeatureExtractor, MinhasArifFeatureExtractor e VarellaLimaFeatureExtractor que montam os conjuntos de características de cada método.

O Projeto da Classificação prevê a construção de quatro classificadores, ChenYuClassifier, YuChenClassifier, MinhasArifClassifier e VarellaLimaClassifier que são auxiliados pelas classes PNN, que implementa uma PNN, a classe KNN que implementa o algoritmo KNN, a classe FFBNN que implementa uma rede neural FFBNN com o auxílio da classe BFGS que implementa o método de BFGS.

6 *Desenvolvimento da Implementação*

Este capítulo descreve o desenvolvimento da implementação dos métodos de classificação de batidas de ECG propostos por este trabalho. Baseando-se no já apresentado projeto de implementação, descrito no capítulo anterior, primeiramente serão descritos detalhes sobre a implementação da fase de detecção de batidas, que como já se sabe, é comum para os métodos propostos. Posteriormente, como feito no projeto, serão descritos detalhes sobre o desenvolvimento da implementação das fases de extração de características e classificação.

A linguagem utilizada pelos autores dos métodos foi Matlab, entretanto para este trabalho a linguagem escolhida foi Java. A linguagem Matlab, muito poderosa para uso acadêmico, além de não ser de gratuita não é portável, com isso a escolha de Java ocorreu devido a facilidade de portar a implementação em computadores pessoais com diferentes sistemas operacionais, bem como em celulares do tipo smartphones. Esta portabilidade é desejada para que os métodos implementados sejam disponíveis em diversas plataformas, inclusive na Web. Mesmo assim, utilizou-se o Matlab para validar os cálculos, ferramentas e algoritmos utilizados.

6.1 Desenvolvimento da Detecção de Batidas

Nesta seção serão apresentados fragmentos do código desenvolvido em Java para a implementação da fase de detecção de batidas de eletrocardiograma oriundas do banco de dados MIT-BIH Arrythmia Database, descrito no capítulo anterior.

Nas figuras 6.1, 6.2, 6.3 e 6.4 temos respectivamente fragmentos do código final das classes Beat, BeatType, Lead e Record que implementam os atributos e as operações propostas pelo projeto estabelecido no capítulo anterior. A figura 6.1 contém os atributos necessários para estruturar uma Batida de ECG, uma lista de amostras do tipo double, intitulada samplesList, que são extraídas diretamente do arquivos de sinais presente no banco de dados, o tempo de duração desta batida, o tipo BeatType no qual a batida foi anotada no arquivo de anotações, o intervalo RR entre a batida atual e a predecessora, bem como o intervalo RR da batida predecessora..

```java
/**
 * Classe que implementa uma estrutura de dados de um QRS
 * contém um vetor de amostras de um intervalo QRS
 * o tempo em que o pico da batida ocorre
 * e o tipo da batida
 */
public class Beat {
    // Lista de amostras
    private double[] samplesList;
    // Tempo de duração da batida
    private double time;
    // Tipo da batida amostrada
    private BeatType type;
    // Derivação da amostra
    private String leadName;
    // Intervalo RR
    private Double RRinterval;
    // Intervalo RR da batida anterior
    private double previousRRInterval;
    // Quantidade de elementos na lista de batidas
    private int listOccupation;
}
```

Figura 6.1: Fragmento da Classe Beat em Java

Já a figura 6.2 contém uma classe java do tipo enumeração, que atribuí valores para os tipos das batidas de ECG que os métodos classificam. Os valores correspondem aos valores possíveis para designar os determinados tipos de batida no arquivo de anotações. Sendo eles: N, L, R, V, A e /. A classe relaciona esses valores com o nome real do tipo da batida (por exemplo: Atrial Premature Beat) e com um valor de enumeração (por exemplo: APB).

```java
public enum BeatType {

    NORMAL("N"), LBBB("L"), RBBB("R"), PVC("V"), APB("A"), PB("/"),
    UNKNOWN("");

    public String toString() {
        switch (this) {
            case NORMAL:
                return "Normal";
            case LBBB:
                return "Left Bundle Branch Block";
            case RBBB:
                return "Right Bundle Branch Block";
            case APB:
                return "Atrial Premature Beat";
            case PB:
                return "Paced Beat";
            case PVC:
                return "Premature Ventricular Contraction";
            case UNKNOWN:
            default:
                return "Unknown";

        }
    }
}
```

Figura 6.2: Fragmento da Classe Beat Type em Java

Na figura 6.3 tem-se uma classe que implementa uma derivação de um exame de ECG, con-

tendo uma lista de batidas, intitulada beatList, do tipo Beat. Além de um nome para a derivação e o baseline desta derivação, valores provenientes dos arquivos de descrição da gravação. A operação setBeatInterval, recebe o valor da posição de uma amostra que corresponde ao pico da onda R, extraído do arquivo de anotações, a lista com as amostras da gravação e o tipo ao qual a amostra é anotada. A partir desta informação 32 amostras predecessoras e 32 sucessoras ao pico são extraídas do arquivo de sinais e colocadas em uma batida, esta por sua vez é inserida na lista de batidas.

```java
/**
 * Clase que implementa uma estrutura de dados de um exame de ECG
 * contém uma lista de batidas de ecg e um identificador
 * @author guilherme
 */
public class Lead {
    public static final int halfInterval = 32;
    private Beat[] beatList;
    private int listOccupation;
    private String name;
    private double base;

    /**
     * Extrai um intervalo de 2*halfinterval do sinal e o coloca em uma lista de batidas
     * @param list Lista de batidas de tipo String
     * @param qrsPeak pico do QRS
     * @param type tipo da batida
     * @param baseline baseline da lead
     * @param leadIndex lead correspondente
     * @return Beat - batida
     */
    public static Beat setBeatInterval(String[][] list, Integer qrsPeak,
                String type, int baseline, int leadIndex, Double RR) {
        ...
    }
}
```

Figura 6.3: Fragmento da Classe Lead em Java

Por fim, a figura 6.4 demonstra o desenvolvimento da classe Record que implementa e estrutura uma gravação de ECG, com uma lista de derivações, denominada leadList, um indicador da ocupação desta lista e um identificador para a gravação. Além é claro da operação que realiza a fase de detecção de batidas propriamente dita, chamada createRecord, que baseada em uma lista de amostras oriundas do arquivo de sinal de ECG da lista de anotações e do descritor da gravação, extraí amostras de ECG, constituindo batidas de ECG baseadas nas batidas anotadas, que por sua vez constituem as derivações que compõem uma gravação.

Vale ressaltar aqui, que para o método proposto por Minhas e Arif é necessária a realização de uma reamostragem dos sinais, que originalmente são de 360 Hz para 250 Hz, esta reamostragem é realizada logo após o download das gravações do banco de dados e por ser realizada apenas para este método ela é considerada como um processo a parte da detecção de batidas. Utiliza-se a função resample do Matlab para realizar a reamostragem.

```java
/**
 * Classe que implementa a estrutura de dados de uma gravação de um ecg
 * Contém derivações com batidas correspondentes
 */
public class Record {
    // Lista de derivações (Lead) que compõem a gravação
    private Lead[] leadList;
    // Assinala o quanto da lista de derivações está preenchida
    private int listOccupation;
    // identificador da gravação
    private int id;

    /**
     * Preenche um record com as informacoes das duas leads que o compoem
     * Cada lead é uma lista de batidas anotadas
     * @param sinalList - lista das batidas
     * @param annList - lista de anotações
     * @param name - nome do arquivo que contem o descritor da gravação
     * @return Record - gravação
     */
    public static Record createRecord(String[][] sinalList, String[][] annList, String name){
        ...
    }
}
```

Figura 6.4: Fragmento da Classe Record em Java

6.2 Desenvolvimento da Extração de Características

Nesta seção será descrito como foi implementada a fase de extração de características dos métodos propostos. Sabe-se que algumas classes projetadas são comuns aos quatro métodos desenvolvidos. A seção começa com a descrição destas estruturas e posteriormente aborda os detalhes da implementação de cada método.

Dois fragmentos de código da classe DWT são visualizados nas figuras 6.5 e 6.6. Na figura 6.5 apenas tem-se as estruturas já propostas pelo projeto desta classe, ou seja, os filtros de cada ondaleta mãe (Symlet 6, Haar e Quadratic Spline) são definidos, além das chamadas paras os dois tipos de transformações possíveis, a transformada discreta de ondaleta realizada por meio do algoritmo piramidal de Mallat (MallatTransform) e a transformada utilizando o algoritmo piramidal à trous (ATrousTransform), que obtém coeficientes de detalhe e aproximação (DWTCoeficients) a partir de um sinal.

Tem-se ainda a figura 6.6 com outras quatro operações que são fundamentais para a realização da transformada. A primeira destas operações é a convolução (conv), que como o nome diz realiza a convolução de um sinal v por um filtro m. A convolução de um sinal de tamanho m por um filtro de tamanho n, produz um novo sinal de tamanho m+n. Sabendo que cada elemento obtido por uma convolução, na verdade é obtido pelo somatório da multiplicação do sinal pelo inverso do filtro. No exemplo da tabela 6.1 temos um sinal de tamanho 8 [1,3,4,2,6,7,8,9] e o filtro de passa-altas Quadratic Spline [2,-2].

A cada três linhas temos os elementos que são multiplicados são aqueles que se interceptam verticalmente e o resultado da soma dos produtos. Por exemplo, nas linhas 4 e 5, temos (1*-

```java
public class DWT {

    // filtro passa baixa para a ondaleta de sym6
    static double[] Sym6LowFilter = {0.015404109327027373, 0.0034907120842174702, -0.11799011114819057,
        -0.048311742585632998, 0.49105594192674662, 0.787641141030194, 0.3379294217276218, -0.072637522786462516,
        -0.021060292512300564, 0.044724901770665779, 0.0017677118642428036, -0.007800708325034148};
    // filtro passa alta para a ondaleta de sym6
    static double[] Sym6HighFilter = {0.007800708325034148, 0.0017677118642428036, -0.044724901770665779,
        -0.021060292512300564, 0.072637522786462516, 0.3379294217276218, -0.787641141030194, 0.49105594192674662,
        0.048311742585632998, -0.11799011114819057, -0.0034907120842174702, 0.015404109327027373};
    // filtro passa baixa para a ondaleta de haar
    static double[] HaarLowFilter = {0.70710678118654757, 0.70710678118654757};
    // filtro passa alta para a ondaleta de haar
    static double[] HaarHighFilter = {-0.70710678118654757, 0.70710678118654757};
    // filtro passa baixa para a ondaleta quadratic spline
    static double[] QSplineLowFilter = {0.125, 0.375, 0.375, 0.125};
    // filtro passa alta para a ondaleta quadratic spline
    static double[] QSplineHighFilter = {2, -2};

    /**
     * Aplica o algoritmo piramidal de Mallat para transformar um sinal
     * @param sigIn sinal
     * @return DWTCoeficients - coeficientes de detalhe e aproximação
     */
    public static DWTCoeficients MallatTransform(double[] sigIn) {
        ...
    }


    /**
     * Aplica o algoritmo piramidal 'a trous' para transformar o sinal
     * @param sigIn sinal
     * @return DWTCoeficients - coeficientes de detalhe e aproximação
     */
    public static DWTCoeficients ATrousTransform(double[] sigIn) {
        ...
    }

```

Figura 6.5: Fragmento da Classe DWT em Java

```java
    /**
     * Realiza a convolução de um vetor v por um filtro m
     * @param m filtro m
     * @param v vetor v
     * @return double[] - vetor convoluido
     */
    public double[] conv(double[] m, double[] v) {
       ...
    }
    /**
     * Extende um sinal centralizando-o e adicionando elementos no final e no começo
     * seguindo um determinado método de insercao
     * @param sigIn sinal a ser extendido
     * @param size tamanho do sinal apos ser extendido
     * @return double[] - sinal extendido
     */
    public static double[] signalExtension(double[] sigIn, int size) {
        ...
    }
    /**
     * Extrai o centro de um sinal extendido
     * @param sigIn sinal a ser extraído
     * @param size tamanho do sinal extraído
     * @return double[] - sinal extraído
     */
    public static double[] signalExtraction(double[] sigIn, int size) {
        ...
    }
    /**
     * Realiza um downsampling que mantem os elementos pares de uma vetor
     * @param v vetor
     * @return double[] - vetor dowsampled
     */
    public static double[] dyadDwsEven(double[] v) {
        ...
    }
}

```

Figura 6.6: Fragmento da Classe DWT em Java

Tabela 6.1: Convolução

	1	3	4	2	6	7	8	9	
-2	2								
= 2									
	1	3	4	2	6	7	8	9	
	-2	2							
= 4									
	1	3	4	2	6	7	8	9	
		-2	2						
= 2									
	1	3	4	2	6	7	8	9	
			-2	2					
= -4									
	1	3	4	2	6	7	8	9	
				-2	2				
= 8									
	1	3	4	2	6	7	8	9	
					-2	2			
= 2									
	1	3	4	2	6	7	8	9	
						-2	2		
= 2									
	1	3	4	2	6	7	8	9	
							-2	2	
= 2									
	1	3	4	2	6	7	8	9	
								-2	2
= -18									
Final:	2	4	2	-4	8	2	2	2	-18

2)+(3*2). O resultado disso é apresentado na linha seguinte e é 4. No final temos os 9 elementos resultantes da convolução, [2,4,2,-4,8,2,2,2,-18]. Outras operações importantes são a operação de extensão de sinal (signalExtension) e a operação de extração de amostras centrais de um sinal (signalExtraction), que se utilizadas antes e após a convolução ajudam a obter um sinal convoluído com o mesmo tamanho do sinal original. Além, da operação de subamostragem diádica (dyaDwsEven), que é semelhante à extração porém extraí apenas as amostras pares do sinal e é utilizada para reduzir uma possível redundância da aplicação dos filtros durante o algoritmo de Mallat.

Nesta seção também são apresentados dois fragmentos da classes MathSupport, no primeiro fragmento presente na figura 6.7 tem-se as operações de média (mean), amplitude relativa (relativeAmplitude), variância (variance) e autocorrelação (autocorrelation). Operações estas necessárias na extração de características dos três métodos propostos e já descritas durante o

```java
public class MathSupport {

    /**
     * Calcula a média de uma lista
     * @param list
     * @return double - média
     */
    public static double mean(double[] list) {
        ...
    }

    /**
     * Calcula a amplitude relativa de um vetor
     * @param list vetor
     * @return double - amplitude relativa
     */
    public static double relativeAmplitude(double[] list) {
        ...
    }

    /**
     * Calcula a variancia de um vetor
     * @param list vetor
     * @return double - variância do vetor
     */
    public static double variance(double[] list) {
        ...
    }

    /**
     * Calcula a autocorrelacao de um vetor
     * @param list vetor
     * @return double[] - vetor de autocorrelação
     */
    public static double[] autocorrelation(double[] list) {
        ...
    }

```

Figura 6.7: Fragmento da Classe MathSupport em Java

projeto da classe.

```java
    /**
     * Computes the number of zero crossings in the given signal
     * @param signal
     * @return
     */
    public static int zeroCrossings(double[] signal) {
        ...
    }

    /**
     * Computes the symmetry value of the given signal
     * @param signal
     * @return
     */
    public static double symmetry(double[] signal) {
        ...
    }
}
```

Figura 6.8: Fragmento da Classe MathSupport em Java

Enquanto que no segundo fragmento presente na figura 6.8 temos duas operações que são utilizadas na extração de características do método de Chen e Yu, são elas a operação zeroCrossings que verifica a quantidade de vezes que o sinal cruza o valor zero e a operação symmetry que verifica o quanto um sinal é simétrico.

As operações realizadas por ambas as classes foram comparadas com as operações análogas implementadas em Matlab. O Matlab/Wavelet Toolbox contém uma implementação do algoritmo de Mallat para a DWT [24]. Os resultados obtidos pela implementação em Java correspondem aos resultados obtidos pela implementação em Matlab tanto para a classe DWT quanto para a classe MathSupport.

6.2.1 Método de Chen e Yu / Método de Varella e Lima

Para finalizar a fase de extração de características destes métodos dois fragmentos de código são apresentados. Na figura 6.9 temos o cabeçalho das operações da classe Cumulants que implementa os cumulantes de segunda, terceira e quarta ordem, seguindo as fórmulas descritas no capítulo 3.

```
1   public double[] cumulant2x(double[] signal) {
2       ...
3   }
4
5   public double[] cumulant3x(double[] signal) {
6       ...
7   }
8
9   public double[] cumulant4x(double[] signal, double[] cumulant2x) {
10      ...
11  }
```

Figura 6.9: Fragmento da Classe Cumulants em Java

Enquanto que na figura 6.10 tem-se o desenvolvimento em Java da classe ChenYuFeatureExtractor proposta no capítulo anterior. Os atributos acabam sendo duas matrizes com as características propostas (trainingSet e testingSet). Tem-se uma operação de extração de características (extractFeatures), que utiliza a operação ATrousTransform da classe DWT, juntamente com as operações da classe Cumulants e das operações variance, zeroCrossings, symmetry, normalizedSummation e informações sobre o intervalo RR da batida para construir o conjunto de características.

O desenvolvimento para o Método de Varella Lima é análogo, e é explicitado na figura 6.11.

Posteriormente, o conjunto pode ser normalizado utilizando a operação featureSetNormalize, que utiliza a função sigmóide tangente hiperbólica para mapear os valores de cada conjunto de características no intervalo [-1,+1].

Uma peculiaridade dos conjuntos obtidos tornou necessária a implementação de uma função de suavização do conjunto, para suavizar discrepâncias nas amplitudes de certas características que podem causar problemas na normalização e com isso na classificação (featureSetSmooth-

ing).

```java
public class ChenYuFeatureExtractor {
    private double[][] trainingSet, testSet;
      /**
     * Extrai as caracteristicas de acordo com o método de Chen e yU
     * @param signal
     * @return double[][] - featuresSet
     */
    public double[][] extractFeatures(Beat[] beat) {
        //DWT
        ...
        //cumulants (3 cumulants for each detail coefficient)
        ...
        //cumulant variances
        ...
        //normalized summation of the cumulants
        ...
        //number of zero crossings
        ...
        //symmetry of the cummulants (only for third and fourth order)
        ...
        //RR interval related features
    }
    /**
     * Função que normaliza um conjunto de vetores de features
     * @param featureSet conjunto de vetores de features
     * @return double[][] conjunto normalizado
     */
    public double[][] feactureSetNormalize(double[][] featureSet) {
        ...
    }
    /**
     * Função que normaliza um conjunto de vetores de features
     * @param featureSet conjunto de vetores de features
     * @return double[][] conjunto normalizado
     */
    public double[][] feactureSetSmoothing(double[][] featureSet) {
      ...
    }
```

Figura 6.10: Fragmento da Classe ChenYuFeatureExtractor em Java

6.2.2 Método de Yu e Chen

O método de Yu e Chen utiliza as classes DWT e MathSupport juntamente com a classe YuChenFeatureExtractor para realizar a fase de extração de características. Na figura 6.12, tem-se um fragmento da implementação desta classe em Java.

Além dos atributos que estruturam o conjunto de treinamento e de teste (trainingSet e testingSet) tem-se três operações. Primeiramente a operação extractFeaturesFS1, que recebe uma derivação (lead), aplica a DWT utilizando a operação MallatTransform com os filtros de Haar, posteriormente são calculadas a variância do sinal e das sub-bandas, a variância da autocorrelação das sub-bandas e a amplitude relativa das sub-bandas.

```java
public class VarellaLimaFeatureExtractor {
    private double[][] trainingSet, testSet;
      /**
       * Extrai as caracteristicas de acordo com o método de Varella e Lima
       * @param signal
       * @return double[][] - featuresSet
       */
    public double[][] extractFeatures(Beat[] beat) {
        //DWT A Trous
        ...
        //cumulantes (segunda ordem D1, terceira e quarta de D2)
        ...
        //variância de S, D1, D2 e A2
        //variância da autocorrelação de D1, D2 e A2
        //amplitude relativa de C2D1 e C4D2
        //variância da autocorrelação de C2D1
        //intervalo RR
    }
    /**
     * Função que normaliza um conjunto de vetores de features
     * @param featureSet conjunto de vetores de features
     * @return double[][] conjunto normalizado
     */
    public double[][] feactureSetNormalize(double[][] featureSet) {
        ...
    }
    /**
     * Função que normaliza um conjunto de vetores de features
     * @param featureSet conjunto de vetores de features
     * @return double[][] conjunto normalizado
     */
    public double[][] feactureSetSmoothing(double[][] featureSet) {
      ...
    }
}
```

Figura 6.11: Fragmento da Classe VarellaLimaFeatureExtractor em Java

Também existe a operação extração de características para a composição do segundo conjunto de características (extractFeaturesFS2), que também aplica a ondaleta de Haar, extraí as mesmas variância e amplitudes relativas, porém extraí o intervalo RR das batidas.

Além das duas operações de extração, tem-se a operação de normalização dos conjuntos de características extraídas (featureSetNormalize) que também utiliza a função sigmóide tangente hiperbólica para mapear os valores de cada conjunto de características no intervalo [-1,+1].

6.2.3 Método de Minhas e Arif

Finalmente a extração das características para o método de Minhas e Arif utiliza as já descritas classes DWT e MathSupport. Além destas classes são necessárias outras duas classes, uma classe que implementa a análise de Componentes Principais, conforma vista no capítulo 3 e a classe MinhasArifFeatureExtractor.

Na figura 6.13 tem-se um fragmento da implementação da classe PCA em Java. A imple-

```java
public class YuChenFeatureExtractor {
    private double[][] trainingSet, testSet;

    /**
     * Função geradora do vetor de features FS1
     * @param Lead lead
     * @return vetor de features
     */
    public static double[] extractFeacturesFS1(Lead lead) {
        // DWT
        //variancia do sinal
        //variancia das sub-bandas
        //variancia da autorrelacao das sub-bandas
        //amplitude relativa das sub-bandas
    }
    /**
     * Função geradora do vetor de features FS2
     * @param Lead lead
     * @return vetor de features  RR)
     */
    public static double[] extractFeacturesFS2(double[] signal){
         //DWT
         //variancia do sinal
         //variancia das sub-bandas
         //variancia da autorrelacao das sub-bandas
         //amplitude relativa das sub-bandas
    }
    /**
     * Função que normaliza um conjunto de vetores de features
     * @param featureSet conjunto de vetores de features
     * @return double[][] conjunto normalizado
     */
    public static double[][] feactureSetNormalize(double[][] featureSet) {
    }
}
```

Figura 6.12: Fragmento da Classe YuChenFeatureExtractor em Java

mentação segue o embasamento teórico proposto no capítulo 3 e o projeto feito no capítulo anterior. Em relação a isto tem-se quatro operações, a operação principal é a PCA que extraí cinco componentes principais para cada um dos vetores existentes em um conjunto de características. Para realizar a extração destes componentes a operação precisa do auxílio de três outras operações, a operação que constrói a matriz de covariância do conjunto de características (covarianceMatrix), a operação que constrói uma matriz de autovetores (eigenvectorMatrix) e uma matriz com autovalores (eigenvalueMatrix), a partir da matriz de covariância.

Enquanto que a figura 6.14 apresenta o fragmento do desenvolvimento da classe MinhasArifFeatureExtractor, com os atributos trainingSet e testingSet, que designam a estrutura dos conjuntos de caraterísticas. Bem como duas operações para a extração das características que diferem entre si pelo uso da Analise de Componentes Principais. A operação extractFeaturesWithoutPCA recebe uma derivação (lead), aplica a DWT utilizando a operação ATrous-Transform com os filtros de Quadratic Spline,posteriormente calculada a variância do sinal e das sub-bandas, a variância da autocorrelação das sub-bandas, a amplitude relativa das sub-

```java
/**
 * Monta a matriz de covariância do conjunto de características
 * @param signal
 * @return
 */
public double[][] covarianceMatrix(double[][] matrix) {
 ...
}
/**
 * Monta a matriz de autovetores de acordo com o algoritmo de decomposição de autovalores
 * @param signal
 * @return
 */
public double[][] eigenvectorMatrix(double[][] matrix) {
 ...

}
/**
 * Monta a matriz de autovalores a partir da matriz de autovetores
 * @param signal
 * @return
 */
public double[][] eigenvalueMatrix(double[][] matrix) {
 ...
}
/**
 * Calcula os 5 componentes principais para cada vetor de caraterísticas
 * @param signal
 * @return
 */
public double[][] PCA(double[][] featSet) throws IOException {
    //matrix de covariância
    //matriz de autovetores
    //matriz de autovalores
}
```

Figura 6.13: Fragmento da Classe PCA em Java

bandas e o intervalo RR.

Enquanto que a operação extractFeaturesWithPCA no final da extração de características realiza ainda o PCA. A classe ainda implementa uma operação de normalização (featureSet-Normalize) utilizando novamente a função tangente sigmóide hiperbólica.

6.3 Desenvolvimento da Classificação

Nesta seção são apresentados alguns fragmentos pontuais que visam elucidar um pouco o desenvolvimento da implementação da fase de classificação dos três métodos.

6.3.1 Método de Chen e Yu / Método de Varella e Lima

A implementação da fase de classificação dos método de Chen e Yu e de Varella e Lima basearam-se no uso de quatro classes, respectivamente, uma classe que implementa a rede neural FFBNN, uma classe que implementa o método BFGS e outras duas que utilizam as duas classes anteriores para classificar as características das batidas de ECG (ChenYuClassifier e VarellaLimaClassifier).

```java
public class MinhasArif {
    private double[][] trainingSet, testSet;
    /**
     * Função geradora do vetor de features sem PCA
     * @param Lead lead
     * @return vetor de features
     */
    public static double[] extractFeacturesWithoutPCA(Lead lead) {
        //DWT
        //variancia do sinal
        //variancia das sub-bandas
        //variancia da autorrelacao das sub-bandas
        //amplitude relativa das sub-bandas
        //intervalo RR
    }
    /**
     * Função geradora do vetor de features com pCA
     * @param Lead lead
     * @return vetor de features  RR
     */
    public static double[] extractFeacturesWithPCA(Lead lead){
         //DWT
         //variancia do sinal
         //variancia das sub-bandas
         //variancia da autorrelacao das sub-bandas
         //amplitude relativa das sub-bandas
         //intervalo RR
         //normalização
         //PCA
    }
   /**
     * Função que normaliza um conjunto de vetores de features
     * @param featureSet conjunto de vetores de features
     * @return double[][] conjunto normalizado
     */
    public static double[][] feactureSetNormalize(double[][] featureSet) {
    }
}
```

Figura 6.14: Fragmento da Classe MinhasArifFeatureExtractor em Java

Na figura 6.15 tem-se a operação que realiza o treinamento da rede neural FFBNN cujos atributos são os mesmo dos propostos pelo projeto no capítulo anterior. Explicando novamente o treinamento, cada entrada do conjunto de treinamento é propagada através da camada de entrada da rede, camada que propaga as saídas de seus neurônios para os neurônios da camada oculta. Essa propagação se dá através da multiplicação do valor de entrada de cada neurônio da camada oculta com o respectivo peso do neurônio. A soma destes produtos por sua vez é à entrada de uma função de ativação que obtém um valor de ativação para cada neurônio da camada oculta. Os valores dos neurônios da camada de saída são calculados com o mesmo procedimento dos neurônios da camada oculta, entretanto, desta vez o valor final de ativação do neurônio é comparado como o valor desejado que está anotado junto ao conjunto de treinamento. A diferença entre o valor de ativação e o desejado é o erro para esta amostra que posteriormente é retro propagado para efetuar a correção dos pesos dos neurônios em todas as camadas. Esse processo é realizado durante um determinado número de épocas de treinamento. Na figura 6.16

```java
public void train() {
        //train the network
        for (int j = 0; j <= this.numberOfEpochs; j++) {
            double RMSerror = 0.0;
            for (int patNum = 0; patNum < trainingSet.nPatterns; patNum++) {
                //propagaFW -> erro
                propagateForward(patNum);
                RMSerror += computeOutputError(patNum);
                //BP
                backpropagateError();
                adjustWeightsBFGS();
            }
            System.out.println("epoch = " + j + "  RMS Error = " + RMSerror);
            if (RMSerror < THRESHOLD) {
                break;
            }
        }
    }
```

Figura 6.15: Fragmento da Classe FFBNN em Java

tem-se a operação que realiza a fase de testes, utiliza a rede previamente treinada, propagando as amostras do conjunto de teste e verificando se elas são classificadas corretamente.

```java
public double[] test(double[][] patterns, double[][] targets) {
        double[] outputs;
            for (int i = 0; i < patterns.length; i++) {
                double output = this.forwardPropagate(patterns[i])[0];
                }
            outputs[i]= output;
            }
        return outputs;
      }

```

Figura 6.16: Fragmento da Classe FFBNN em Java

Os resultados obtidos pela implementação em Java para o método de Chen e Yu não foram satisfatórios, ficaram muito aquém dos resultados propostos pelo método e durante a fase de treinamento utilizada, percebeu-se que a implementação acabava não convergindo e caindo em erros mínimos locais, o que motivou uma outra abordagem. Como a fase de treinamento é a fase mais importante e mais complexa de uma rede neural, resolveu-se utilizar a implementação do treinamento de uma rede neural em Matlab, com estruturas mais otimizadas para o treinamento, obtendo assim os pesos de uma rede neural e com estes pesos realizar a fase de teste em Java. O código em Matlab é demonstrado na figura 6.17.

No algoritmo feito, utiliza-se a função newff que cria uma estrutura de rede neural FFBNN, com 60 neurônios na camada oculta e 6 na camada de saída, funções de ativação tangente sigmóide e o uso de método BFGS (trainbfg) para a correção dos pesos. Posteriormente, os pesos obtidos após o treinamento (train) são escritos em arquivos que por sua vez são carregados pelo código em Java para a realização dos testes em Java. Esta abordagem foi utilizada tanto

```
% 3.1 Treinamento Teste
[f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12,f13,f14,f15,f16,f17,f18,f19,f20,f21,f22,f23,f24,f25,f26,f27,f28,f29,f30,classe1,classe2,classe3,classe4,classe5,classe6] =
 textread('input\chenYuTrainingOut6.txt','%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f');

entradaTreinamento = [f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12,f13,f14,f15,f16,f17,f18,f19,f20,f21,f22,f23,f24,f25,f26,f27,f28,f29,f30];
saidaTreinamento = [classe1,classe2,classe3,classe4,classe5,classe6];

net = newff(minmax(entradaTreinamento'),[60 6],{'tansig','tansig'},'trainbfg');

net.trainParam.epochs = 10000;
net.trainParam.min_grad = 7e-6;

net = train(net,entradaTreinamento',saidaTreinamento');

%Escreve os pesos encontrados no treinamento:
peso1 = net.IW{1};
peso2 = net.LW{2};
peso3 = net.b{1};
peso4 = net.b{2};

[of1,of2,of3,of4,of5,of6,of7,of8,of9,of10,of11,of12,of13,of14,of15,of16,of17,of18,of19,of20,of21,of22,of23,of24,of25,of26,of27,of28,of29,of30,oclasse1,oclasse2,oclasse3
 = textread('input\chenYuTestingOut6.txt','%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f');

entradaTeste = [of1,of2,of3,of4,of5,of6,of7,of8,of9,of10,of11,of12,of13,of14,of15,of16,of17,of18,of19,of20,of21,of22,of23,of24,of25,of26,of27,of28,of29,of30];
saidaTeste = [oclasse1,oclasse2,oclasse3,oclasse4,oclasse5,oclasse6];

simulacaoTeste = sim(net,entradaTeste');
csvwrite('lmchenYuOut.csv', simulacaoTeste')
csvwrite('lmchenYuOut2.csv', saidaTeste)
csvwrite('lmchenYuWeightInput.csv', peso1)
csvwrite('lmchenYuWeightOutput.csv', peso2)
csvwrite('lmchenYuWeightBias1.csv', peso3)
csvwrite('lmchenYuWeightBias2.csv', peso4)

```

Figura 6.17: FFBNN em Matlab

para o método de Chen e Yu quanto para o método de Varella e Lima.

6.3.2 Método de Yu e Chen

A fase de classificação do método de Yu e Chen conta com duas classes, uma classe que implementa uma rede neural probabilística (PNN) e uma classe que utiliza operações da rede para realizar a classificação. Nas figuras 6.18 e 6.19 temos fragmentos da classe PNN onde é explicitado o desenvolvimento das suas três principais operações.

```
public static int pnnGaussianClassify(double sigma, double[] sample, SetForPNN trainingSet) throws IOException {
        int outNeuron = -1;
        double largest = 0.0;
        double[] sumNeuron = new double[classes.length];
        //CAMADA DE SOMATÓRIO
        //6 neurônios(sum)
        sumNeuron[0] = gaussianPatternLayer(sample, trainingSet.nSet, sigma);
        sumNeuron[1] = gaussianPatternLayer(sample, trainingSet.lSet, sigma);
        sumNeuron[2] = gaussianPatternLayer(sample, trainingSet.rSet, sigma);
        sumNeuron[3] = gaussianPatternLayer(sample, trainingSet.vSet, sigma);
        sumNeuron[4] = gaussianPatternLayer(sample, trainingSet.aSet, sigma);
        sumNeuron[5] = gaussianPatternLayer(sample, trainingSet.pSet, sigma);
        //CAMADA DE SAÍDA
                //camada de saída com apenas um neurônio ou valor de saída (outNeuron)
        for (int i = 0; i < classes.length; i++) {
            if (sum[i] > largest) {
                largest = sum[i];
                outNeuron = i;
            }
        }
        return outNeuron;
    }
```

Figura 6.18: Fragmento da Classe PNN em Java

O comportamento destas operações é o seguinte. A camada de entrada (presente em pnnGaussianClassify) apenas distribui as amostras para os neurônios da camada de padrão (operação gaussianPatternLayer), sendo que todas as entradas são propagadas para todos os neurônios da camada de padrão. Cada amostra do conjunto de treinamento corresponde a uma unidade da camada de padrão, e cada unidade de padrão (operação gaussianKernel) estima a função de

densidade de probabilidade da entrada ser parecida com a amostra do conjunto de treinamento na unidade e consequentemente ser classificada com a mesma classe da amostra do conjunto de treinamento. Cada classe que o classificador classifica corresponde a um neurônio na camada de somatório (presente em pnnGaussianClassify) que recebe a saída de todas as unidades de padrão cuja amostra de treinamento corresponde a classe referente a essa unidade. A probabilidade acumulada pela unidade de somatório é então transmitida para a camada de saída (presente em pnnGaussianClassify) que decide a qual dentre as classes suportadas a amostra pertence.

```
1  public static double gaussianKernel(double[] sample, double[] classSample, double sigma) throws IOException {
2          double ret = 0.0;
3          double[] diff = vectorDifference(sample, classSample);
4          double product = vectordotMultiply(diff, diff);
5          //valor de cada kernel gaussiano na camada de padrão
6          ret = Math.exp((-1 * product) / (2 * (sigma * sigma)));
7          return ret;
8
9      }
10 public static double gaussianPatternLayer(double[] sample, double[][] classSet, double sigma) throws IOException {
11         double ret = 0.0;
12         //CAMADA DE PADRÃO
13         //cada "unit" é um neurônio da cada de padrão
14         for (int i = 0; i < classSet.length; i++) {
15             double unit = 0.0;
16             double[] classSample = classSet[i];
17             unit = gaussianKernel(sample, classSample, sigma);
18             ret += unit;
19         }
20         return ret / classSet.length;
21     }
```

Figura 6.19: Fragmento da Classe PNN em Java

O Matlab não possui a PNN programada em sua Neural Network Toolbox, por isso, programou-se uma PNN muito semelhante em Matlab que obteve resultados semelhantes aos obtidos com a implementação em Java.

6.3.3 Método de Minhas e Arif

O método de Minhas e Arif utiliza apenas duas classes para o desenvolvimento da fase de classificação, uma classe para gerenciar a classificação (MinhasArifClassifier) e a classe KNN que implementa o algoritmo KNN. A figura 6.20 contém o principal fragmento de código da classe KNN, que elucida o funcionamento deste algoritmo.

Seu funcionamento é demonstrado pelas seguintes observações. A distância Euclidiana no procedimento euclidianDistance é definida como a distância entre dois pontos, sendo expressa como a raiz quadrada do quadrado da diferença entre os pontos. O procedimento sort apenas ordena os valores do vetor de distâncias em ordem crescente. O procedimento mode verifica entre as k menores distâncias, a classe mais frequente, ou seja, a classe moda.

Novamente, o Matlab não possui a KNN programada. Assim, programou-se uma KNN, muito semelhante à programada em Java, em Matlab que obteve resultados semelhantes aos obtidos com a implementação em Java.

```
1   public class Elemento {
2    public double d;
3    public BeatType classe;
4    public Elemento(double dt, BeatType it) {
5        d = dt;
6        classe = it;
7    }
8 }
9    public static int KNN(double[] sample, SetForKNN tSet, int k, int l) {
10       int ret = -1;
11       Elemento[] euclidianDistance = new Elemento[tSet.bigSet.length];
12       int[] kNearestNeighbors = new int[k];
13       // calcula a distancia euclidiana de sample para todas as amostras do conjunto de treinamento
14       for (int i = 0; i < tSet.bigSet.length; i++) {
15           Elemento aux = new Elemento(euclidianDistance(sample, tSet.bigSet[i]), tSet.classSet[i]);
16           euclidianDistance[i] = aux;
17       }
18               //ordena as distancias obtidas
19       QuickSort.sort(euclidianDistance);
20       //encontra a classe mais frequente entre as k amostras mais próximas de sample
21       for (int i = 0; i < kNearestNeighbors.length; i++) {
22           kNearestNeighbors[i] = euclidianDistance[i].classe.ordinal();
23       }
24       ret = mode(kNearestNeighbors, l);
25
26       return ret;
27   }
28
29
```

Figura 6.20: Fragmento da Classe KNN em Java

6.4 Resumo

Os métodos foram implementados em Java seguindo o projeto do capítulo anterior e sendo validados com implementações semelhantes em Matlab. Para a detecção de batidas a operação createRecord resume o funcionamento da fase, pois que baseada em uma lista de amostras oriundas do arquivo de sinal de ECG da lista de anotações e do descritor da gravação, extraí amostras de ECG, constituindo batidas de ECG baseadas nas batidas anotadas, que por sua vez constituem as derivações que compõem uma gravação. Além disso, para o método proposto por Minhas e Arif foi necessária a realização de uma reamostragem dos sinais, que originalmente são de 360 Hz para 250 Hz utilizando a função resample do Matlab.

A fase de extração de características tem nas classes FeatureExtractor as operações que resumem o funcionamento desta fase, a extração de características utiliza a classe DWT, juntamente com as operações da classe MathSupport para construir o conjunto de características, os métodos de Varella e Lima e Chen e Yu ainda utilizando a Classe Cumulants enquanto que o método de Minhas e Arif ainda utiliza a classe PCA.

A fase de classificação é focada no comportamento dos classificadores, bem sucedidos para Minhas e Arif e Yu e Chen. Entretanto, os resultados obtidos pela implementação em Java para o método de Chen e Yu não foram satisfatórios, ficaram muito aquém dos resultados propostos pelo método. Resolveu-se então utilizar a implementação do treinamento de uma rede neural em Matlab, com estruturas mais otimizadas para o treinamento, obtendo assim os pesos de uma rede neural e com estes pesos realizar a fase de teste em Java. A mesma abordagem acabou

sendo depois utilizada para o método de Varella e Lima.

7 *Testes e Resultados da Implementação*

Este capítulo especifica os testes realizados para verificar o comportamento dos métodos implementados por este trabalho. Ainda, apresenta os resultados obtidos pelas implementações propostas, bem como uma breve discussão destes resultados.

7.1 Teste Propostos

Para avaliar o comportamento dos métodos implementados foram utilizados 23 gravações do banco de dados de arritmias do MIT/BIH [9], que contém diversos eletrocardiogramas anotados com diversos tipos de arritmias. Foram utilizadas as mesmas amostras para realizar os testes de todos os métodos implementados a fim de evitar qualquer superestimativa.

Tabela 7.1: As gravações selecionadas do banco de dados do MIT/BIH.

Tipo	Gravação MIT/BIH	# Treinamento	# Teste
N	103, 113, 115, 123, 220, 234	600	600
LBBB	109, 111, 207, 214	600	600
RBBB	118, 124, 212, 231	600	600
PVC	119	200	200
	221	150	150
	200, 233	400	400
APB	209	150	150
	222	100	100
	232	600	600
PB	107, 217	600	600
Total		11600	11600

A Tabela 7.1 mostra detalhes das gravações do banco de dados do MIT/BIH que foram utilizadas para construir os conjuntos de treinamento e de teste. Como as redes neurais precisam de conjuntos grandes de treinamento para atingir um bom desempenho, foram utilizadas 11.600 instâncias para cada conjunto. Vale enfatizar que todos os métodos utilizam os mesmos conjuntos de batidas.

Os testes realizados para cada um dos métodos foram efetuados com a seguinte metodologia:

- Detecção das batidas - através dos arquivos do banco de dados e seguindo os valores da tabela 7.1.
- Extração das características - Utilizando as técnicas propostas por cada um dos métodos.
- Classificação das características - Utilizando o classificador proposto pelo método.

Para o método de Minhas e Arif, realizou-se testes variando o valor k, que determina o número de vizinhos considerados, do classificador. Enquanto que para o método de Yu e Chen variou-se o coeficiente de suavização, σ. Para o método de Chen e Yu e de Varella e Lima não foram variadas características, entretanto as redes foram treinadas 20 vezes e posteriormente testadas.

7.2 Resultados

Nesta seção são apresentados os resultados para cada método com as respectivas variações nos valores de σ, k ou ainda nos pesos da FFBNN.

7.2.1 Método de Chen e Yu

Como o método de treinamento da Rede Neural FFBNN é não-determinístico, ou seja, a cada treinamento obtêm-se pesos diferentes para a mesma topologia de rede, realizou-se 20 treinamentos, em Matlab. Por fim utilizou-se o conjunto de pesos que proporcionou maior acurácia na classificação do conjunto de testes, realizada em Java. Na tabela 7.2 tem-se a matriz de confusão dos resultados obtidos com a classificação do conjunto de testes que obtiveram maior acurácia.

Tabela 7.2: Matriz de Confusão para o Método de Chen e Yu

Classe	NORMAL	LBBB	RBBB	PVC	APB	PB
NORMAL	3424	33	25	17	41	60
LBBB	18	2264	25	41	20	32
RBBB	21	23	2258	44	18	36
PVC	11	18	15	1084	9	13
APB	16	6	4	8	804	12
PB	16	7	6	10	11	1150

A soma dos valores de cada linha da tabela designa a quantidade de batidas cada tipo anotadas no conjunto de testes, para cada coluna temos a quantidade de batidas classificadas para cada tipo. Observando a linha NORMAL, temos que das 3600 batidas anotadas como NORMAL, 3424 batidas foram classificadas como NORMAL, 33 como LBBB, 25 como RBBB, 17 como APB, 41 como PVC e 60 como PB. Totalizando uma especificidade de 95,11%. Na linha seguinte tem-se que das 2400 batidas anotadas como LBBB, 2264 batidas foram classificadas como LBBB, 18 como NORMAL, 25 como RBBB, 41 como APB, 20 como PVC e 32 como PB. Totalizando uma sensitividade de 94,33%. Posteriormente, tem se uma sensitividade de 94,09% para RBBB, sensitividade de 94,25% para PVC, sensitividade de 94,58% para APB e sensitividade de 95,83% para PB. Finalmente a acurácia total do método foi de 94,7%.

Os valores obtidos pelo conjunto de pesos mais acurado para este método acabou sendo 5% menor do que os valores apresentados na sua publicação.

7.2.2 Método de Varella e Lima

De forma semelhante à utilizada no Método de Chen e Yu, realizou-se 20 treinamentos da rede FFBNN em Matlab. Por fim utilizou-se o conjunto de pesos que proporcionou maior acurácia na classificação do conjunto de testes, realizada em Java. Na tabela 7.3 tem-se a matriz de confusão dos resultados obtidos com a classificação do conjunto de testes que obtiveram maior acurácia.

Tabela 7.3: Matriz de Confusão para o Método de Varella e Lima

Classe	NORMAL	LBBB	RBBB	PVC	APB	PB
NORMAL	3599	0	0	1	0	0
LBBB	0	2389	2	7	0	2
RBBB	0	2	2398	0	0	0
PVC	0	2	2	1145	0	1
APB	0	0	0	0	850	0
PB	0	0	0	1	0	1199

Observando a linha NORMAL, temos que das 3600 batidas anotadas como NORMAL, 3599 batidas foram classificadas como NORMAL e apenas uma batida foi classificada como PVC. Totalizando uma especificidade de 99,94%. Na linha seguinte tem-se que das 2400 batidas anotadas como LBBB, 2389 batidas foram classificadas como LBBB, apenas duas como RBBB, 7 como APB e 2 como PB. Totalizando uma sensitividade de 99,54%. Posteriormente, tem-se uma sensitividade de 99,83% para RBBB, sensitividade de 99,48% para PVC, sensitividade de 99,88% para APB e sensitividade de 99,92% para PB. Finalmente a acurácia total do método foi de 99,83%.

7.2.3 Método de Minhas e Arif

Para o método de Minhas e Arif variou-se o valor k, que determina o número de vizinhos considerados para a classificação com os valores [1,2,3,4,5,10,50]. Os resultados para o conjunto de características sem PCA encontra-se na tabela 7.4 enquanto que na 7.5 encontram-se os valores com PCA.

Tabela 7.4: Resultados para o conjunto sem PCA e diferentes valores de k

k	Especificidade(%)	Sensitividade(%)					Acurácia(%)
		LBBB	RBBB	PVC	APB	PB	
1	99,81	98,29	99,04	97,91	98,00	99,67	99,00
2	99,83	98,63	98,75	97,91	96,35	98,67	98,79
3	99,75	98,17	99,04	98,26	97,88	99,25	98,94
4	99,75	98,42	99,00	98,17	97,29	98,83	98,89
5	99,69	98,04	98,96	97,91	98,00	98,83	98,81
10	99,72	97,71	98,63	97,39	97,29	98,67	98,56
50	99,31	95,42	97,54	94,17	92,24	98,00	96,97

Para o conjunto de característica sem a utilização de PCA a melhor acurácia obtida foi de 99% considerando apenas um vizinho, ou seja, k=1.

Tabela 7.5: Resultados para o conjunto com PCA e diferentes valores de k

k	Especificidade(%)	Sensitividade(%)					Acurácia(%)
		LBBB	RBBB	PVC	APB	PB	
1	99,58	98,33	98,75	97,91	97,06	99,50	98,79
2	99,83	98,75	98,00	97,65	95,29	98,58	98,55
3	99,69	98,25	98,67	97,91	97,88	99,25	98,83
4	99,75	98,33	98,38	98,26	97,53	98,83	98,77
5	99,67	98,17	98,63	97,65	97,88	98,58	98,70
10	99,69	97,96	98,08	97,04	97,29	98,33	98,42
50	99,33	96,42	96,92	93,83	92,00	97,67	96,97

Enquanto que para o conjunto de característica com a utilização de PCA a melhor acurácia obtida foi de 98,83% considerando três vizinhos, ou seja, k=3.

Tabela 7.6: Matriz de Confusão para o conjunto de características sem PCA e $k = 1$

Classe	NORMAL	LBBB	RBBB	PVC	APB	PB
NORMAL	3593	1	5	0	1	0
LBBB	2	2359	17	9	13	0
RBBB	6	8	2377	2	6	1
PVC	0	13	2	1126	7	2
APB	1	4	5	7	833	0
PB	0	1	0	3	0	1196

A matriz de confusão para o conjunto de características sem PCA e k=1 encontra-se na tabela 7.6 enquanto que na 7.7 encontra-se a matriz para o conjunto com PCA e k=3.

Tabela 7.7: Matriz de Confusão para o conjunto de características com PCA e $k = 3$

Classe	NORMAL	LBBB	RBBB	PVC	APB	PB
NORMAL	3589	1	9	0	1	0
LBBB	4	2358	16	10	12	0
RBBB	17	7	2368	1	6	1
PVC	0	11	2	1126	9	2
APB	2	3	5	8	832	0
PB	0	1	1	7	0	1191

Comparados aos resultados obtidos pelo artigo original, os resultados deste trabalho apresentam um comportamento semelhante conforme se aumenta o valor de k. Além disso, a acurácia para o conjunto sem PCA é apenas 0,49% menor e com PCA 0,64% menor.

7.2.4 Método de Yu e Chen

Para o método de Yu e Chen variou-se o valor k, que determina o coeficiente de suavização, σ,no intervalo [0,1 .., 1,0]. Os resultados para o conjunto de características FS1 encontra-se na tabela 7.8 enquanto que na 7.9 encontram-se os valores para o FS2.

Tabela 7.8: Resultados para o FS1 com diferentes valores de σ

σ	Especificidade(%)	Sensitividade(%)					Acurácia(%)
		LBBB	RBBB	PVC	APB	PB	
0,10	99,19	95,92	96,38	91,39	93,76	99,83	96,83
0,20	98,31	89,13	94,25	81,22	90,17	99,75	93,47
0,30	97,00	79,46	91,08	78,35	88,59	99,75	89,97
0,40	91,97	72,17	82,88	77,65	86,47	99,75	84,97
0,50	80,58	67,00	69,33	76,52	85,29	99,83	77,38
0,60	80,50	62,50	56,79	74,09	85,06	99,75	73,56
0,70	80,58	57,33	47,33	72,00	87,88	99,75	70,56
0,80	80,64	52,63	41,75	70,61	90,00	99,75	68,47
0,90	80,69	47,75	37,79	69,91	91,88	99,75	66,72
1,00	80,75	43,33	35,63	69,39	94,12	99,58	65,47

Tanto para o conjunto FS1 quanto para o conjunto FS2 a melhor acurácia foi obtida com $\sigma = 0,1$, totalizando 96,83% para o FS1 e 98,67% para o FS2.

A matriz de confusão para o conjunto de características FS1 e $\sigma = 0,1$ encontra-se na tabela 7.10 enquanto que na 7.11 encontra-se a matriz para o conjunto FS2 e $\sigma = 0,1$.

Tabela 7.9: Resultados para o FS2 com diferentes valores de σ

σ	Especificidade(%)	Sensitividade(%)					Acurácia(%)
		LBBB	RBBB	PVC	APB	PB	
0,10	99,50	97,58	97,79	98,87	98,82	99,83	98,67
0,20	98,92	93,67	96,29	98,09	97,29	99,83	97,18
0,30	98,00	88,88	94,50	89,39	97,65	99,83	94,70
0,40	96,22	86,38	90,08	82,17	98,47	99,75	92,05
0,50	85,44	84,33	78,46	82,00	99,53	99,67	85,93
0,60	80,81	82,13	69,00	81,83	99,76	99,75	82,09
0,70	80,83	79,54	62,17	81,30	99,88	99,75	80,10
0,80	80,97	76,96	55,83	81,04	100,00	99,75	78,28
0,90	81,06	73,75	49,75	80,78	100,00	99,75	76,36
1,00	81,11	68,75	44,50	79,57	100,00	99,75	74,14

Tabela 7.10: Matriz de Confusão para o conjunto de características FS1 e $\sigma = 0,1$

Classe	NORMAL	LBBB	RBBB	PVC	APB	PB
NORMAL	3571	7	12	0	10	0
LBBB	0	2302	17	52	29	0
RBBB	22	30	2313	15	20	0
PVC	1	35	21	1051	42	0
APB	2	40	10	1	797	0
PB	0	1	0	1	0	1198

Tabela 7.11: Matriz de Confusão para o conjunto de características FS2 e $\sigma = 0,1$

Classe	NORMAL	LBBB	RBBB	PVC	APB	PB
NORMAL	3582	2	13	0	3	0
LBBB	3	2342	16	13	26	0
RBBB	11	20	2347	4	16	2
PVC	1	5	3	1137	4	0
APB	1	3	5	1	840	0
PB	0	1	0	1	0	1198

De modo semelhante ao método de Minhas e Arif, os resultados deste trabalho se comparados aos resultados obtidos pelo artigo original, apresentam um comportamento semelhante conforme se aumenta o valor de σ. Além disso, a acurácia para o conjunto FS2 é apenas 0,98% menor e para o FS1 2,49% menor.

7.3 Discussão

A Tabela 7.12, onde (A) é acurácia, (E) é especificidade e (S) é sensitividade, apresenta uma comparação da implementação dos quatro métodos em Java. O método de Varella e Lima supera os demais métodos tanto na taxa de acurácia,com 99.83%, quanto na taxa de sensitividade mais

Tabela 7.12: Resultados dos testes com a implementação dos métodos

Medidas	Minhas e Arif		Yu e Chen		Chen e Yu	Varella e Lima
	Sem PCA	Com PCA	FS1	FS2		
(A) (%)	99.0	98.79	98.67	96.83	94.7	99.83
(E) (%)	99.81	99.58	99.5	99.19	95.11	99.94
(S) LBBB (%)	98.29	98.33	97.58	95.92	94.33	99.54
(S) RBBB (%)	99.04	98.75	97.79	96.38	94.09	99.83
(S) PVC (%)	97.91	97.91	98.87	91.39	94.25	99.48
(S) APB (%)	98.00	97.06	98.82	93.76	94.58	99.88
(S) PB (%)	99.67	99.50	99.83	99.83	95.83	99.92

baixa. É importante considerar a taxa de sensitividade mais baixa para garantir que o método possui um desempenho consistente e possua boa capacidade de discriminação para todos os tipos de batidas, evitando que patologias importantes sejam percebidas na classificação.

Como pode ser visto, os resultados não condizem com os propostos pelos autores e apresentados na Tabela 4.2. Os experimentos com as implementações dos três outros métodos neste trabalho foram bem sucedidos em um computador, atingindo uma acurácia de 99% para o método de Minhas e Arif sem PCA, 98.79% com PCA, 98,67% para o método de Yu e Chen usando o FS2, 96.83% com o FS1 e apenas 94,7% para o método de Chen e Yu. Entretanto, a acurácia do método de Minhas e Arif sem PCA ficou 0,49% menor e 0,8% menor com PCA, e a do método de Yu e Chen ficou 0,98% menor para o FS2 2,49% menor, já o método de Yu e Chen ficou 5% menor. Como foi realizada uma implementação em Java e outro análoga em Matlab, a fim de validar a primeira, a discrepância observada com os valores propostos pelos autores pode ter relação com o uso de estruturas mais otimizadas pelos autores.

Entretanto, utilizando as mesmas estruturas básicas destes métodos foi possível obter um conjunto de características e um classificador com um desempenho maior do que inclusive os valores alegados pelos autores em suas publicações, e esta fato corrobora para a validade da implementação realizada.

8 *Conclusões e Trabalhos Futuros*

Este capítulo apresenta as conclusões obtidas com a realização deste trabalho.

8.1 Conclusões

Neste trabalho, foi apresentado um estudo comparativo dos métodos de classificação de batidas de eletrocardiograma destinados a detectar arritmias. Este estudo foi motivado pela necessidade de uma resposta com boa acurácia para a análise de exames de ECG em diversas situações, como antes de um paciente ter alta precipitada e ainda em situações mais graves, quando uma resposta automática pode antecipar o resgate de um paciente. Em relação aos objetivos, pode-se dizer que este trabalho realizou a análise teórica e prática com sucesso e ainda auxilou no desenvolvimento de um novo método de classificação de arritmias.

Em relação à literatura especializada existente, foram selecionados três métodos que reportaram altas acurácias juntamente com altas sensibilidades, nominalmente: Método de Chen e Yu, de Minhas e Arif e de Yu e Chen. Os resultados obtidos por estes métodos não corresponderam os resultados alegados nas suas respectivas publicações. Além deste três métodos foi apresentado o método desenvolvido pelo mestrado de Fernando Varella com o auxílio deste trabalho, o Método de Varella e Lima.

Novamente, a implementação do método de Chen e Yu, que utiliza um conjunto composto por 30 características, extraídas utilizando DWT, cumulantes e estatísticas básicas, classificadas por meio de uma FFBNN teve um resultado 5% inferior ao reportado na sua publicação, 94,7% da implementação deste trabalho contra 99,7% proposto pelos autores. Implementou-se este método em Matlab (da mesma forma que os autores) e em Java, para as duas implementações os resultados não foram satisfatórios.

A implementação do método de Minhas e Arif, que utiliza por sua vez DWT e estatísticas básicas em um de seus conjuntos de características e PCA, DWT, estatísticas básicas em outro, classificados pelo algoritmo KNN, obteve resultados mais próximos aos propostos. Para

o conjunto sem PCA obteve 99% de acurácia, apenas 0,49% a menos do que o alegado na sua publicação. Configurando-se o melhor resultados dentre os três métodos existentes na literatura.

O segundo melhor resultado foi atingido pelo método de Yu e Chen, com 98.83% de acurácia, para um conjunto de características que baseia-se em DWT e estatísticas básicas, classificado por uma PNN. Entretanto os resultados de nenhum dos métodos existentes na literatura foram superiores ao método de Varella e Lima.

Com isso, um dos objetivo deste trabalho foi realizado com sucesso, pois após a discussão e implementação de quatro métodos distintos foi possível indicar que o método de Varella e Lima é o mais apto para classificar batidas de ECG com alta acurácia e sensitividades.

Sabendo que a aplicação direta dos métodos é um sistema de suporte ao diagnóstico de cardiopatias, há um requisito forte de que o método possua bom desempenho na classificação das batidas. No método de Varella e Lima, estatísticas de ordem superior obtidas dos coeficientes da transformada de ondaleta do sinal das batidas, aliados a medidas básicas de estatística, como variância, amplitude relativa e autocorrelação, formam a base para representar batidas de ECG em um conjunto de características.

Cada uma das batidas passa por esse processo de extração de características para que seja transformada em um conjunto de onze características, as quais serão utilizadas como entrada para o processo de classificação. Para efetuar a classificação foi definida uma rede neural do tipo MLP, a qual é capaz de identificar até seis tipos de arritmias. Essas características levaram o método a alcançar uma taxa 99,83% de acurácia na classificação das batidas. Entretanto, os resultados mais importantes foram as altas taxas de sensitividade alcançadas pelo método, pois elas foram superiores a 99,48% para todos os tipos de batidas, mostrando a grande capacidade de discriminação do método.

8.2 Trabalhos Futuros

Como trabalho futuro é planejado estender o método para que seja possível classificar mais de seis tipos de arritmias sem prejudicar o seu desempenho nos seis tipos de batidas já contemplados, de modo que um sistema ainda mais robusto de monitoramento remoto de pacientes possa ser implantado.

Referências Bibliográficas

[1] BRASIL. *Comentários sobre Estatísticas da Saúde: Assistência Médico-Sanitária 2009*. [S.l.], 2009. Disponível em: http://www.ibge.gov.br/home/estatistica/populacao/condicaodevida/ams/2009/comentarios.pdf.

[2] BRASIL. *Estatísticas da Saúde: Assistência Médico-Sanitária 2009*. [S.l.], 2009. Disponível em: http://www.ibge.gov.br/home/estatistica/populacao/condicaodevida/ams/2009/ams2009.pdf.

[3] BRASIL. *Portaria n.º 1101/GM*. [S.l.], Junho 2002. Disponível em: http://dtr2001.saude.gov.br/sas/PORTARIAS/Port2002/Gm/GM-1101.htm.

[4] World Health Organization. *Estimated deaths per 100,000 population by cause, and Member State*. [S.l.], 2004.

[5] FISCH, C. Centennial of the string galvanometer and the electrocardiogram. *Journal of the American College of Cardiology*, 2000.

[6] AUGUSTYNIAK, P.; TADEUSIEWICZ, R. *Ubiquitous cardiology : emerging wireless telemedical applications*. [S.l.]: Information Science Reference, 2009.

[7] CRILEY, D. *Normal Heart Conduction System*. Disponível em: www.blaufuss.org.

[8] DESCONHECIDO. *Derivações Precordiais: V1, V2, V3, V4, V5 e V6*. Disponível em: http://www.cvphysiology.com/Arrhythmias/A013c.htm.

[9] AGATELLER, A. *Schematic diagram of normal sinus rhythm for a human heart as seen on ECG (with English labels)*. Disponível em: http://en.wikipedia.org/wiki/File:SinusRhythmLabels.svg.

[10] DESCONHECIDO. *Cálculo da Frequência Cardíaca*. Disponível em: http://www.ambulancetechnicianstudy.co.uk/rhythms.html.

[11] DESCONHECIDO. *Arritmia Sinusal, Marcapasso Migratório e Fibrilação Atrial*. Disponível em: [http://allaboutim.webs.com/apps/blog/show/5364432-premature-ventricularcontraction.

[12] MALLAT, S. G. A theory for multiresolution signal decomposition: The wavelet representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 11, n. 7, p. 674–693, jul 1989.

[13] OSOWSKI, S.; LINH, T. H. Ecg beat recognition using fuzzy hybrid neural network. *IEEE Transactions on Biomedical Engineering*, v. 48, 2001.

[14] PARZEN, E. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 1962.

[15] FLETCHER, R. *Practical methods of optimization*. [S.l.: s.n.], 1980. 38 p.

[16] ENGIN, M. Ecg beat classification using neuro-fuzzy network. *Pattern Recognition Letters*, v. 25, p. 1715–1722, 2004.

[17] GÜLERA, I.; ÜBEYLI, E. D. Ecgbeat classifier designed by combined neural network model. *Pattern Recognition*, v. 38, p. 199–208, 2005.

[18] YU, S.-N.; CHEN, Y.-H. Electrocardiogram beat classification based on wavelet transformation and probabilistic neural network. *Pattern Recognition Letters*, v. 28, p. 1142–1150, 2007.

[19] YU, S.-N.; CHOU, K.-T. A switchable scheme for ecg beat classification based on independent component analysis. *Expert Systems with Applications*, v. 33, p. 824–829, 2007.

[20] CHEN, Y.-H.; YU, S.-N. Subband features based on higher order statistics for ecg beat classification. In: *Proceedings of the 29th Annual International Conference of the IEEE EMBS*. [S.l.: s.n.], 2007.

[21] MINHAS, F. ul A. A.; ARIF, M. Robust electrocardiogram (ecg) beat classification using discrete wavelet transform. *Physiological Measurement*, v. 29, p. 555–570, 2008.

[22] KHADTARE, M. S.; SAHAMBI, J. Ecg arrhythmia analysis by multicategory support vector machine. In: *Applied Computing*. [S.l.]: Springer Berlin / Heidelberg, 2004.

[23] MOODY, G. B. *WFDB Applications Guide*. 2011. Disponível em: `http://www.physionet.org/physiotools/wag/wag.htm`.

[24] MATHWORKS. *Product Documentation: Single-level discrete 1-D wavelet transform.* [S.l.]. Disponível em: `http://www.mathworks.com/help/toolbox/wavelet/ref/dwt.html`.