UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

André Furlan

Caracterização de *voice spoofing* para fins de verificação de locutores com base na transformada wavelet e na análise paraconsistente de características

São José do Rio Preto, SP
2020

André Furlan

Caracterização de *voice spoofing* para fins de verificação de locutores com base na transformada wavelet e na análise paraconsistente de características

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Orientador: Prof. Dr. Rodrigo Capobianco Guido

São José do Rio Preto, SP
2020

André Furlan

Caracterização de *voice spoofing* para fins de verificação de locutores com base na transformada wavelet e na análise paraconsistente de características

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Orientador: Prof. Dr. Rodrigo Capobianco Guido

Comissão Examinadora

Professor Dr. Rodrigo Capobianco Guido
UNESP - Campus de São José do Rio Preto
Orientador

Professora Dra. Renata Spolon Lobato
UNESP - Campus de São José do Rio Preto

Professor Dr. Aleardo Manacero Júnior
UNESP - Campus de São José do Rio Preto

São José do Rio Preto, SP
2020

AGRADECIMENTOS

A todos e todas que vieram antes de nós e que desenvolveram a ciência e a tecnologia possibilitando os avanços da humanidade, sem a dedicação desses e dessas gigantes nada disso seria possível.
A todos os trabalhadores e trabalhadoras, que são os reais motores da nossa sociedade, que um dia o conhecimento os liberte de suas amarras.
A minha Mãe que, mesmo com dificuldades, conseguiu criar seus filhos para o mundo.
A meu orientador, que com competência e gentileza me ajudou até aqui.

E especialmente a minha esposa e companheira que me incentiva aos estudos e vibra com nossas conquistas.

RESUMO

Voice spoofing é uma técnica que tenta ludibriar sistemas de segurança baseados em identificação por voz. Este trabalho visa inicialmente, através da decomposição do sinal com wavelets e posterior cálculo da energia em intervalos baseados na escala BARK ou MEL, determinar qual a melhor combinação BARK/MEL-wavelet para que se obtenha uma separação máxima entre duas classes (voice spoofing e audio original) usando análise paraconsistente de características. Após apuração da melhor combinação, os vetores de características gerados são submetidos a ensaios para classificação mudando-se o tamanho do conjunto de treinamento e testes de forma que, para cada novo teste, esses conjuntos são misturados de forma aleatória. Os classificadores usados foram os de distância Euclidiana e Manhattan além de Máquina de suporte de vetores(SVM). A acurácia máxima conseguida na distância Euclidiana e Manhattan foi de 90,97% e usando SVM 95,12%.

Palavras-chave: voice spoofing, wavelet, análise paraconsistente.

ABSTRACT

Voice spoofing is a technique that attempts to circumvent security systems based on voice identification. This work initially aims, through the decomposition of the signal with wavelets and later calculation of the energy in intervals based on the BARK or MEL scale, to determine which is the best BARK/MEL-wavelet combination to obtain a maximum separation between two classes (voice spoofing and audio) using paraconsistent feature analysis. After determining the best combination, the feature vectors generated are subjected to tests for classification, changing the size of the training set and tests so that, for each new test, these sets are mixed randomly. The classifiers used were those of Euclidean and Manhattan distance as well as Support vector machine(SVM). The maximum accuracy achieved at Euclidean and Manhattan distance was 90.97% and using SVM 95.12%.

Key-words: voice spoofing, wavelet, paraconsistent analysis.

Lista de Figuras

Lista de Tabelas

Sumário

SUMÁRIO

Capítulo 1

Introdução

1.1 Considerações Iniciais

Os *voice spoofing attacks* do tipo *playback speech* constituem o tema de estudo deste trabalho.

1.2 Objetivos

Neste trabalho para dissertação de mestrado a intenção é encontrar um conjunto de características que demonstrem ser as mais disjuntas possíveis para fins de separação entre as classes, com o objetivo de melhorar a acurácia de classificadores para detecção de ataques de *voice spoofing*. Caraterísticas essas com base na transformada *wavelet*, devido a sua boa resolução em relação às dimensões de tempo e frequência. Essas características serão avaliadas usando a análise paraconsistente de acordo com o trabalho [Gui19] recentemente publicado.

1.3 Estrutura do trabalho

No capítulo 2 se realizará uma revisão dos seguintes conceitos:

- Engenharia paraconsistente.

- Filtros digitais usando wavelets.
- Caracterização dos processos de produção da voz humana.
- Amostragem, quantização, entre outros.

Em seguida no capítulo 3 apresentar-se-á a proposta de aproximação do problema usada nesta investigação.

No capítulo 4 são mostrados os testes e resultados obtidos para, no capítulo 5 se apresentar as conclusões.

Ao final do documento se encontram as referências usadas.

Capítulo 2

Revisão de Bibliográfica

2.1 Conceitos utilizados

2.1.1 Engenharia paraconsistente de características

Dentro do processo de classificação frequentemente surge a questão:
Os vetores de características criados proporcionam uma boa separação de classes?

O método de cálculo do plano paraconsistente é uma ferramenta que pode ser usada para responder essa questão.

O processo inicia-se após a aquisição dos vetores de características para cada classe (C_n) onde n é o índice de cada uma delas. Se o número de classes presentes for, por exemplo, quatro então estas poderão ser representadas por C_1, C_2, C_3, C_4.

Em seguida será necessário o cálculo de duas grandezas:

- A menor similaridade intraclasse (α).
- A razão de sobreposição interclasse (β)

α indica o quanto de similaridade os dados têm entre si dentro de uma mesma classe, β mostra a razão de sobreposição entre diferentes classes. Idealmente α deve ser maximizada e β minimizada para um desempenho ótimo dos classificadores.

Inicialmente é necessária a normalização dos vetores de características de forma que a soma de todos os seus valores seja um.

Em seguida a obtenção de α se dá selecionando-se os maiores e os menores valores de cada uma das posições de todos os vetores de características de cada classe gerando assim um vetor para os valores maiores e outro para os menores.

O **vetor de similaridade**(svC_n) é obtido fazendo-se a diferença item a item dos maiores em relação aos menores.

Finalmente e para cada classe é tirada a média dos valores de cada vetor de similaridade, α é o menor valor dentre essas médias.

A figura 2.1 ilustra este processo.

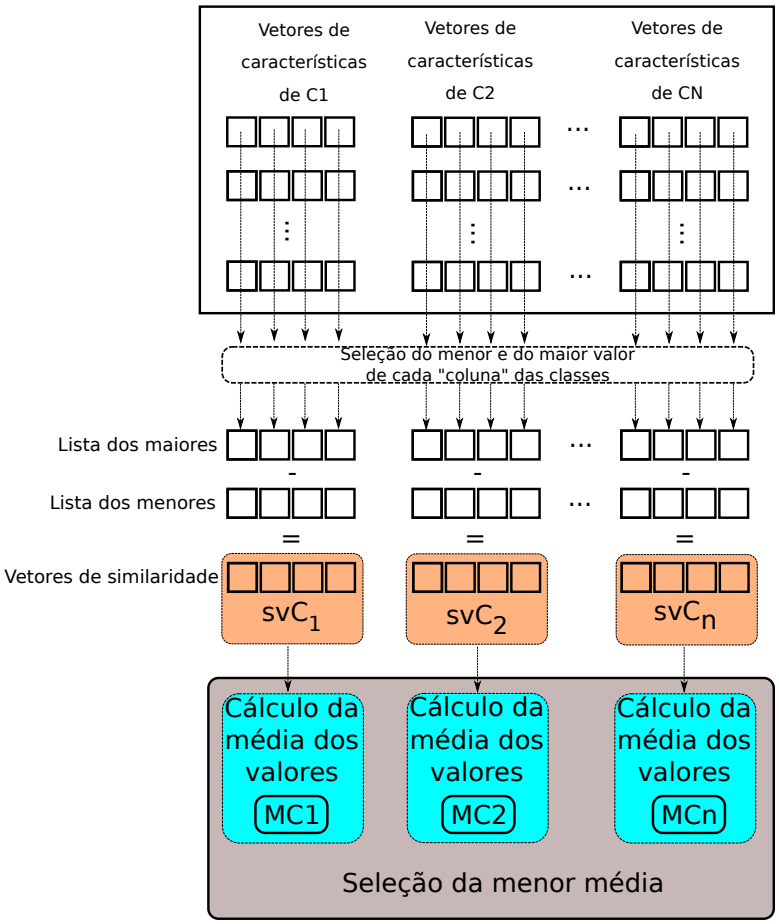


Figura 2.1: Cálculo de α

A obtenção de β, assim como ilustrado na figura 2.2, também se dá selecionando-se os maiores e os menores valores de cada uma das posições de todos os vetores de características de cada classe gerando assim um vetor para os valores maiores e outro

para os menores.

Na sequência se segue com o cálculo de R cujo valor é a quantidade de vezes que um valor do vetor de características de uma classe se encontra no intervalo de valores maiores e menores de outra classe.

É necessário o cálculo de F que é o número máximo de sobreposições possíveis entre classes e é dado por:

$$F = N.(N-1).X.T \tag{2.1}$$

onde:

- N é a quantidades de classes.
- X é quantidade de vetores de características por classe.
- T é o tamanho do vetor de características.

Finalmente, β é calculado:

$$\beta = \frac{R}{F} \tag{2.2}$$

Nesse ponto é importante notar que $\alpha = 1$ sugere fortemente que os vetores de características de cada classe são similares e representam suas respectivas classes precisamente. Complementarmente $\beta = 0$ sugere os vetores de características de classes diferentes não se sobrepõe [Gui19].

- Verdade → Fé total ($\alpha = 1$) e nenhum descrédito ($\beta = 0$)
- Ambiguidade → Fé total ($\alpha = 1$) e descrédito total ($\beta = 1$)
- Falsidade → Fé nula ($\alpha = 0$) e descrédito total ($\beta = 1$)
- Indefinição → Fé nula ($\alpha = 0$) e descrédito total ($\beta = 0$)

No entanto, raramente α e β terão tais valores, na maioria do tempo $0 \leqslant \alpha \leqslant 1$ e $0 \leqslant \beta \leqslant 1$, por isso, se torna necessário o cálculo do **grau de certeza**(G_1) e do **grau de contradição**(G_2).

$$G_1 = \alpha - \beta \tag{2.3}$$

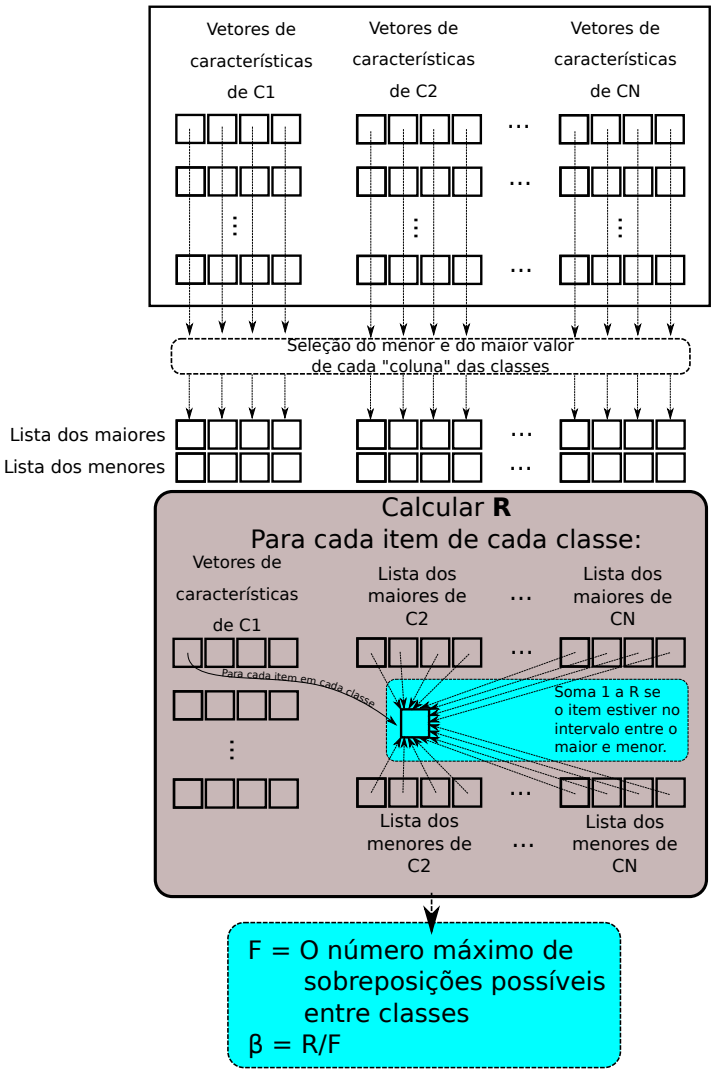


Figura 2.2: Cálculo de β

$$G_2 = \alpha + \beta - 1 \qquad , \tag{2.4}$$

onde: $-1 \leqslant G_1$ e $1 \geqslant G_2$.

Os valores de G_1 e G_2 em conjunto definem os graus entre verdade e falsidade, ou seja, $G_1 = -1$ e $G_1 = 1$ respectivamente e também os graus entre indefinição e ambiguidade, ou seja, $G_2 = -1$ e $G_2 = 1$ respectivamente.

O plano paraconsistente para fins de visualização e maior rapidez na avaliação dos resultados como ilustrado na figura 2.3 tem quatro cantos definidos:

- (-1,0) → Falsidade.

- (1,0) → Verdade.
- (0,-1) → Indefinição.
- (0,1) → Ambiguidade.

Perceba que na figura 2.3 existe um pequeno círculo, este indica onde se encontram as classes nos graus explicados da listagem anterior.

Para se ter ideia em que área exatamente se encontram as classes avaliadas se deve calcular as distâncias(D) do ponto $P = (G_1, G_2)$ dos limites supracitados. Tal cálculo pode ser feito da seguinte forma:

$$D_{-1,0} = \sqrt{(G_1 + 1)^2 + (G_2)^2} \quad , \tag{2.5}$$

$$D_{1,0} = \sqrt{(G_1 - 1)^2 + (G_2)^2} \quad , \tag{2.6}$$

$$D_{0,-1} = \sqrt{(G_1)^2 + (G_2 + 1)^2} \quad , \tag{2.7}$$

$$D_{0,1} = \sqrt{(G_1)^2 + (G_2 - 1)^2} \quad , \tag{2.8}$$

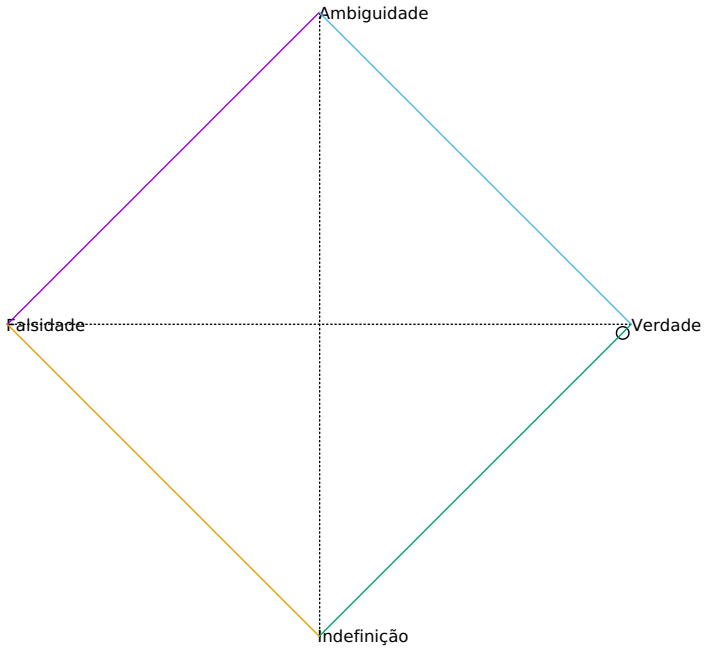


Figura 2.3: O plano paraconsistente.

Na prática, ou seja, para fins de classificação geralmente considera-se a distância em relação ao ponto *"(1,0) → Verdade"* para se dizer os quão separados são os vetores de características analisados: Quanto mais próximo o círculo estiver de *(1,0)* mais disjuntos eles serão.

2.1.2 Filtros digitais wavelets

Filtros digitais *wavelet* vem para suprir as deficiências de janelamento de sinal apresentadas pelas transformadas de Fourier e pelas transformadas curtas de Fourier. *Wavelets* contam com variadas funções filtro e tem tamanho de janela variável o que permite uma análise multirresolução [AWG09].

As *wavelets* proporcionam a análise do sinal de forma detalhada tanto no espectro de baixa frequência quanto no de alta frequência.

É importante observar que, quando se trata de transformada *wavelet* seis elementos estão presentes: dois filtros de análise, dois filtros de síntese e as funções ortogonais *scaling* e *wavelet*. No tocante a sua aplicação só a transformada direta, e não a inversa, será usada na construção dos vetores de características, portanto, os filtros de síntese, a função *scaling* e a função *wavelet* não serão elementos abordados aqui, pois, esses só interessariam caso houvesse a necessidade da transformada inversa.

No contexto dos filtros digitais baseados em *wavelets* o tamanho da janela recebe o nome de **suporte**. Janelas definem o tamanho do filtro que será aplicado ao sinal, quando esse é pequeno se diz que a janela tem **um suporte compacto** [P$^+$96].

Se diz que uma *wavelet* tem boa **resposta em frequência** quando, na aplicação da mesma na filtragem das frequências não são causadas muitas pertubações indesejadas ao sinal, as wavelets de Daubechies se destacam neste quesito por serem *maximamente planas* (Maximally-flat) nos platôs de resposta em frequência como indicado na figura 2.4.

Além da resposta em frequência a aplicação de um filtro digital *wavelets* também pode gerar o que se chama de **resposta em fase**, esse deslocamento pode ser **linear**, **quase linear** ou **não linear**.

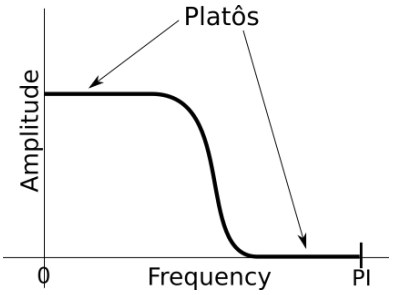


Figura 2.4: Platôs maximamente planos em um filtro digital

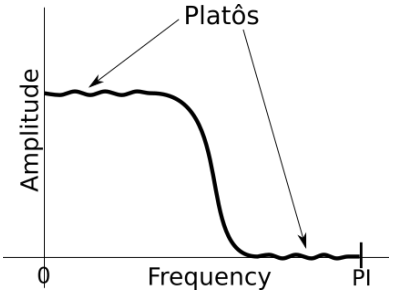


Figura 2.5: Platôs não maximamente planos de um filtro digital

- Na resposta em fase **linear** há o mesmo deslocamento de fase para todos os componentes do sinal.
- Quando a resposta em fase é **quase linear** existe uma pequena diferença no deslocamento dos diferentes componentes do sinal.
- Finalmente, quando a resposta é **não linear** acontece um deslocamento significativamente heterogêneo para as diferentes frequências formantes do sinal.

Idealmente é desejável que todo filtro apresente boa resposta em frequência e resposta em fase linear.

Wavelet	**Resposta em frequência**	**Resposta em fase**
Haar	Pobre	Linear
Daubechies	Quanto maior o suporte, melhor. *Maximally-flat*	Não linear
Symmlets	Quanto maior o suporte, melhor. Não *Maximally-flat*	Quase linear
Coiflets	Quanto maior o suporte, melhor. Não *Maximally-flat*	Quase linear

Tabela 2.1: Algumas wavelets mais populares e suas propriedades

2.1.3 Sub-amostragem ou *Downsampling*

O termo sub amostragem ou *downsampling* se refere ao processo de diminuir o tamanho do sinal após a aplicação de um filtro digital, em *wavelets* isso se dá excluindo-se alternadamente os itens do vetor que representa o sinal como mostrado na figura 2.6.

Na figura 2.6 as partes pretas contêm dados e as brancas representam os elementos removidos.

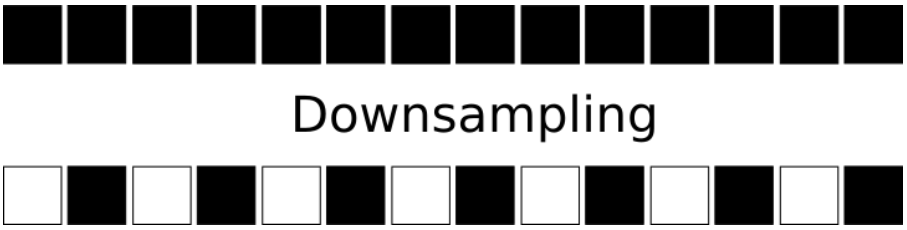


Figura 2.6: Sub-amostragem

O algoritmo de Malat para transformada wavelet

O algoritmo de Malat torna aplicação das *wavelets* no sinal em uma simples multiplicação de matrizes, o sinal que deve ser transformado se torna uma matriz linear vertical já os filtros passa-baixa e passa-alta tornam-se, nessa ordem, linhas de uma matriz quadrada que será completada segundo regras que serão mostradas mais adiante.

É importante que essa matriz quadrada tenha de aresta a mesma quantidade de itens que o nosso sinal, ou seja, se o sinal tem quatro elementos então a matriz de filtros deve ser uma de 4x4.

Algo interessante a se notar é que, para que seja possível a transformada wavelet, basta ter disponível o vetor do filtro passa-baixa calculado a partir da *mother wavelet*(que é a função geradora desse filtro) já que o filtro passa-alta pode ser construído a partir da ortogonalidade do primeiro.

Determinar a ortogonal de um vetor significa construir um vetor, tal que, o produto escalar do vetor original com sua respectiva ortogonal tenha como resultado zero.

Considere $h[\cdot]$ como sendo o vetor do filtro passa baixa e $g[\cdot]$ seu correspondente ortogonal. Sendo assim temos $h[\cdot].g[\cdot] = 0$,.

Portanto, se $h[\cdot] = [a, b, c, d]$ então seu ortogonal será $g[\cdot] = [d, -c, b, -a]$ pois:

$$h[\cdot].g[\cdot] = 0 \tag{2.9}$$

$$h[\cdot].g[\cdot] = [a, b, c, d].[d, -c, b, -a] = 0 \tag{2.10}$$

$$h[\cdot].g[\cdot] = (a.d) + (b. - c) + (c.b) + (d. - a) = 0 \tag{2.11}$$

$$h[\cdot].g[\cdot] = a.d\breve{}b.c + c.b\breve{}d.a = 0 \tag{2.12}$$

$$h[\cdot].g[\cdot] = a.d\breve{}a.d + b.c\breve{}b.c = 0 \tag{2.13}$$

$$h[\cdot].g[\cdot] = 0 + 0 = 0 \tag{2.14}$$

A título de exemplo considere:

- O filtro passa baixa baseado na wavelet Haar: $h[\cdot] = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]$,
- E seu respectivo valor ortogonal: $g[\cdot] = [\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}]$,
- Considere também o seguinte sinal: $sinal = [1, 2, 3, 4]$.

Se o tamanho do sinal a ser tratado é quatro, ou seja, o sinal tem quatro pulsos, e se pretende-se aplicar o filtro Haar, a seguinte matriz é construída:

$$\begin{pmatrix} \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0 \\ \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 0, 0 \\ 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \\ 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \\ \end{pmatrix} \tag{2.15}$$

No entanto, filtros Haar tem apenas dois valores e, necessariamente, a linha da matriz deve ter quatro itens. Para resolver este problema basta completar cada uma das linhas com zeros. A matriz é montada de forma que a mesma seja ortogonal.

Montada a matriz de filtros segue-se com os cálculos da transformada:

$$\begin{pmatrix} \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0 \\ \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 0, 0 \\ 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \\ 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} = \begin{pmatrix} \frac{3}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \\ \frac{7}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \tag{2.16}$$

Realizada a multiplicação é necessário agora montar o sinal filtrado, isso é feito escolhendo, dentro do resultado, valores alternadamente de forma que o vetor resultante seja:

$$resultado = \left[\frac{3}{\sqrt{2}}, \frac{7}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}\right] \quad . \tag{2.17}$$

Perceba que na transformada wavelet mostrada em 2.15, 2.16 e 2.17 a **aplicação dos filtros sobre o vetor ocorreu apenas uma vez**, sendo assim, se diz que o sinal recebeu a **transformada wavelet nível 1**. A cada transformada há uma separação do sinal em dois componentes: O de baixa e o de alta frequência.

É possível aplicar mais de um nível de transformadas ao sinal, para que se possa fazer isso a transformada wavelet nível 2 deve considerar apenas a parte de baixa frequência da primeira transformada, já transformada de nível 3 deve considerar apenas a parte de baixa frequência da transformada nível 2 e assim consecutivamente.

Nos exemplos numéricos mostrados nas tabelas 2.2, 2.3 e 2.4 usou-se um filtro haar cujos valores são $\frac{1}{2}$ e $-\frac{1}{2}$. Os dados destacados em **verde** correspondem ao **vetor original** que será tratado, cada uma das linhas são os resultados das transformadas wavelets nível 1, 2, 3 e 4 respectivamente, as partes em **azul** correspondem a porção de **baixa frequência**, as partes em **amarelo** correspondem as porções de **alta frequência**.

Perceba que na tabela 2.2, a partir da transformada nível 2, apenas as partes de baixa frequência são modificadas.

Isso implica que, no momento da implementação do algoritmo de Malat **para níveis maiores que 1** a abordagem será **recursiva**, em outras palavras, a partir do nível 1 se deve aplicar Malat apenas às metades de baixa frequência geradas pela transformação anterior.

Sinal	32	10	20	38	37	28	38	34	18	24	24	9	23	24	28	34
Nível 01	21	29	32,5	36	21	16,5	23,5	31	11	-9	4,5	2	-3	7,5	-0,5	-3
Nível 02	25	34,25	18,75	27,25	-4	-1,75	2,25	-3,75	11	-9	4,5	2	-3	7,5	-0,5	-3
Nível 03	29,62	23	-4,625	-4,25	-4	-1,75	2,25	-3,75	11	-9	4,5	2	-3	7,5	-0,5	-3
Nível 04	26,3125	3,3125	-4,625	-4,25	-4	-1,75	2,25	-3,75	11	-9	4,5	2	-3	7,5	-0,5	-3

Tabela 2.2: Exemplo numérico de wavelet haar aplicada a um vetor

O algoritmo de Malat e a Wavelet packet

Na transformada *wavelet packet* os filtros aplicados são os mesmos e a recursividade da abordagem também é a mesma, no entato, aplicada a transformada nível 1, é necessário que a transformada nível 2 seja aplicada aos componentes de baixa e de alta frequência. Sendo assim a transformada *wavelet packet* obtém um nível de detalhes em todo o espectro de frequência maior do que uma *wavelet* regular.

Os exemplos mostrados nas tabelas 2.3 e 2.4 mostram como se dão as transformações na porção de **baixa** e de **alta** frequência respectivamente após a *wavelet* nível 1.

Devido ao *downsampling* aplicado as porções de alta frequência essas partes acabam por ficar "espelhadas", ou seja, suas sequências ficam invertidas. Para resolver esse problema e preservar a ordem do sinal os filtros são aplicados em ordem inversa nas porções de alta frequência [JlCH01]. Isso altera como o algoritmo de Malat deve ser implementado para a transformada *wavelet packet*, já que dessa vez, é preciso se atentar a ordem da aplicação dos filtros passa-alta e passa-baixa.

Sinal	32	10	20	38	37	28	38	34
Nivel 01	21	29	32,5	36	21	16,5	23,5	31
Nivel 02	25	34,25	18,75	27,25	-4	-1,75	2,25	-3,75
Nivel 03	29,62	23	-4,625	-4,25	-1,125	3	-2,875	-0,75
Nivel 04	26,3125	3,3125	-0,1875	-4,4375	0,9375	-2,0625	-1,0625	-1,8125

Tabela 2.3: Exemplo numérico de packet wavelet haar aplicada a um vetor (Porção da baixa frequência)

Para uma visualização mais completa consulte a figura 6.1 no apêndice deste documento.

Sinal	18	24	24	9	23	24	28	34
Nivel 01	11	-9	4,5	2	-3	7,5	-0,5	-3
Nivel 02	10	1,25	-5,25	1,25	1	3,25	2,25	-1,75
Nivel 03	5,625	-2	4,375	-3,25	-1,125	2	2,125	0,25
Nivel 04	1,8125	3,8125	3,8125	0,5625	0,4375	-1,5625	0,9375	1,1875

Tabela 2.4: Exemplo numérico de packet wavelet haar aplicada a um vetor (Porção da alta frequência)

2.1.4 Amostragem, quantização e o formato do arquivo Wave

Os arquivos no formato *wave*, segundo [Sap19], se estruturam como o ilustrado na figura 2.7.

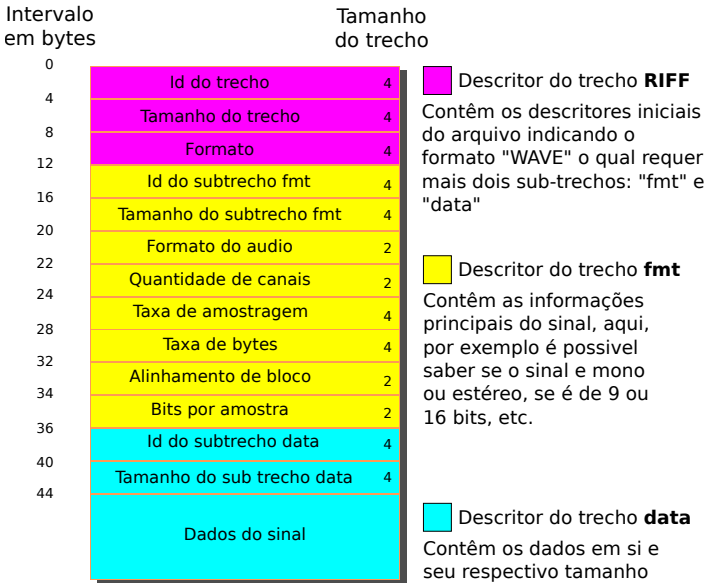


Figura 2.7: Estrutura do arquivo Wave

A estrutura de interesse se localiza na última parte do arquivo, mais especificamente no bloco "data", aqui os dados são organizados como um grande vetor de números, cada um deles, indicando a intensidade do sinal naquele ponto.

2.1.5 Caracterização dos processos de produção da voz humana

A fala possui três grandes áreas de estudo: fisiológica (ou fonética articulatória), acústica (ou fonética acústica) e perceptual (ou comumente chamada percepção da fala)

[KG14].

Neste trabalho o foco será apenas na acústica, já que não serão analisados aspectos da fisiologia relacionada a voz e sim o sinal sonoro propriamente dito.

Vozeada versus não-vozeada

Quando da análise de voz se pode levar em consideração as partes vozeadas e/ou não-vozeadas do sinal. As partes vozeadas são aquelas produzidas com ajuda das pregas vocais, as partes não-vozeadas não tem participação desta estrutura.

Frequência fundamental da voz

Também conhecida como f_0 é o componente periódico resultante da vibração das pregas vocais, em termos de percepção se pode interpretar f_0 como o tom da voz (pitch) [KG14].

Vozes agudas tem um pitch alto, vozes mais graves tem um pitch baixo, a alteração do pitch durante a fala é definido como entonação.

A frequência fundamental da voz é a velocidade na qual uma forma de onda se repete por unidade de tempo, ou seja, o número de ciclos vibratórios produzidos pelas pregas vocais, num segundo, sendo assim, as medidas de f_0 geralmente são apresentadas em Hz [Fre13].

A medição de f_0 está sujeita a contaminações surgidas das variações naturais de *pitch* tipicas da voz humana [Fre13].

A importância de se medir f_0 corretamente vem do fato de que, além de carregar boa parte da informação da fala, f_0 é a base para construção das outras frequências, pois essas são múltiplas de f_0.

Formantes

O primeiro formante (f_1), relaciona-se à amplificação sonora na cavidade oral posterior e à posição da língua no plano vertical; o segundo formante (f_2) à cavidade oral anterior

e à posição da língua no plano horizontal. O terceiro formante (f_3) relaciona-se às cavidades à frente e atrás do ápice da língua; o quarto formante (f_4), ao formato da laringe e da faringe na mesma altura [V+14].

2.1.6 Escalas e energias dos sinais

O cálculo da energia de uma sinal consiste na soma dos quadrados de seus valores dentro de intervalos definidos, após isso, se pode aplicar uma normalização do sinal. As escalas escolhidas são BARK e MEL.

A escala BARK

BARK foi pensada tendo em mente vários tipos de sons. Essa escala corresponde a 25 bandas críticas de audição. Suas frequências base de audiometria são, em Hz: **20, 100, 200, 300, 400, 510, 630, 770, 920, 1080, 1270, 1480, 1720, 2000, 2320, 2700, 3150, 3700, 4400, 5300, 6400, 7700, 9500, 12000, 15500**. Nesses intervalos que se dão os cálculos de energia.

Desconsiderando qualquer etapa intermediária que possa ser adicionada, as energias calculadas nos intervalos definidos na escala BARK podem, por si mesmas ser um vetor de características, como mostrado na figura 2.8.

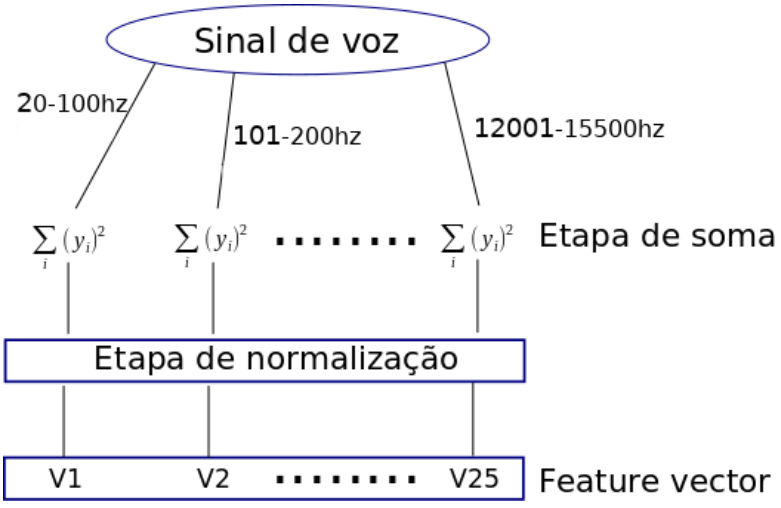


Figura 2.8: Cálculo de vetores de características com BARK

A escala MEL

Escala Mel é uma adaptação da escala Bark para sinais de voz. MEL é mais adaptada a voz humana, pois, foi construída a partir de experimentos com humanos. Nessa escala a bandas consideradas são em Hz: **20, 160, 394, 670, 1000, 1420, 1900, 2450, 3120, 4000, 5100, 6600, 9000, 14000**.

A variante que será usada neste trabalho é conhecida como *Mel-frequency cepstral coefficients*(MFCC) a qual inclui, além dos intervalos definidos, a aplicação da transformada direta cosseno ou análise de componente principal seguida de derivações no vetor do sinal resultante.

Novamente, desconsiderando qualquer etapa intermediária que possa ser adicionada, as energias calculadas nos intervalos definidos na escala MEL podem, por si mesmas ser um vetor de características, como mostrado na figura 2.8.

2.2 Trabalhos correlatos

No artigo [RFLC19] foi apresentado um esquema de diferenciação entre a fala comum e aquela vinda de um dispositivo reprodutor. O foco da análise se dá na distorção causada pelo alto-falante segundo a energia e outras várias características do espectro do sinal. Uma base com 771 sinais de fala foi criada para cada um dos quatro dispositivos de gravação usados totalizando 3084 trechos de áudio. Uma *support vector machine* (SVM) foi usada como classificador. De acordo com os experimentos a *taxa de verdadeiros positivos* é de 98,75% e a *taxa de verdadeiros negativos* é de 98,75%.

Em [YXWW19] é mostrado um método para diferenciar a voz de um locutor verdadeiro da voz gerada por sistemas usando sintetizadores baseados no *modelo oculto de Markov* (HMM). SAS[YJ19] foi a escolha de base de dados. Este método usa coeficientes de características logarítmicos extraídos de wavelets que são apresentados a um classificador SVM. Os resultados obtidos tiverem, em média, mais de 99% de acurácia.

Usando uma decomposição por espalhamento baseada em wavelets e convertendo o resultado em coeficientes cepstrais (SCCs) o artigo [SSAL17] cria um vetor de caracte-

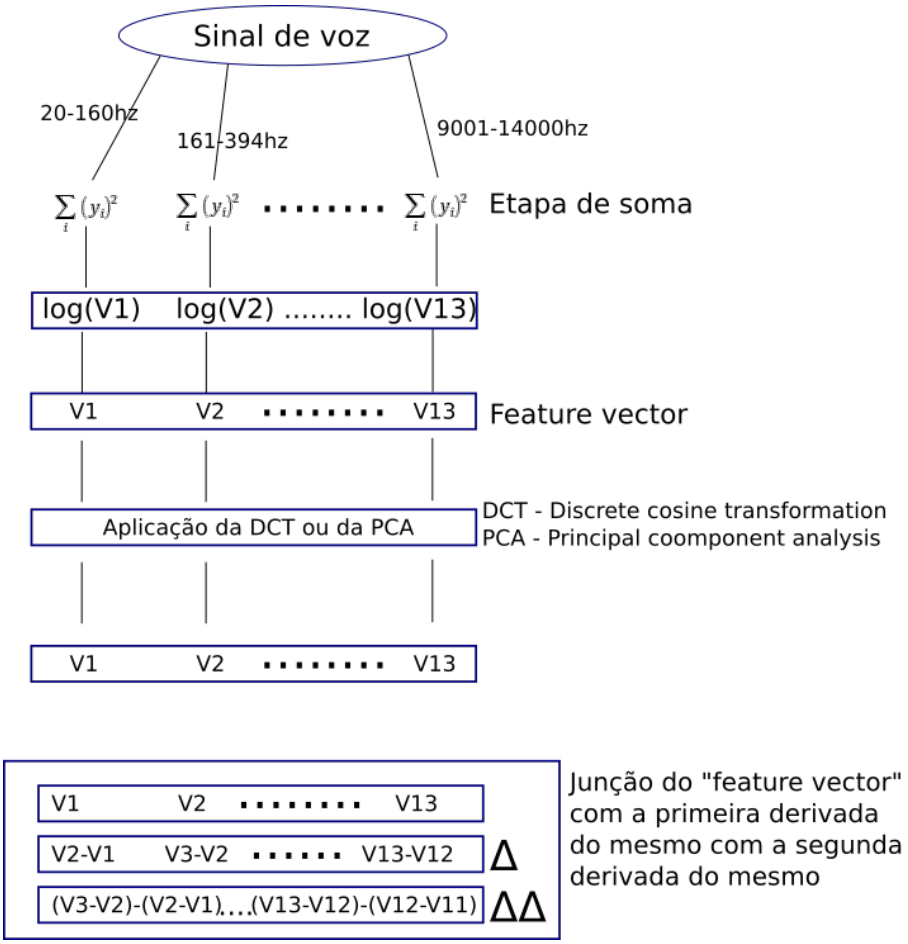


Figura 2.9: Cálculo de vetores de características com MEL

rísticas que é avaliado por modelos de mistura Gaussiana (GMM). SAS e ASVspoof 2015 [ZW15] foram as bases de dados escolhidas para testes. Em relação aos resultados foram usadas a *taxa de falsos verdadeiros* (FAR) que representa a taxa de ocorrências falsas classificadas como verdadeiras e a *taxa de falsos falsos* (FRR) que é a taxa de ocorrências verdadeiras classificadas como falsas. Aos pontos em que FAR é igual a FRR chamou-se de pontos de taxa de erros iguais e a *taxa de erros iguais* (ERR) é o valor de $\frac{FAR}{FRR}$. Considerando isso nos experimentos foi obtida uma ERR geral de 0,18.

Já em [AV19] os autores usam o *"Zero time windowing"* ou janelamento de tempo zero (ZTW), conceito esse que deve ser melhor entendido durante a confecção da dissertação, para, em conjunto com a análise cepstral do espectro gerado, fazer a análise do sinal. Os experimentos foram feitos usando-se a base ASVspoof 2017[TK17] com um classifica-

dor GMM, a taxa geral de ERR dos experimentos foi de 0,1475.

Em [YLYL19] é citado que existe uma diferença entre as propriedades espectrais da voz original e da voz gravada. No escrito são usados coeficientes cepstrais sobre os quais são aplicados uma média e uma normaliza de variância para diminuir o impacto dos ruídos na classificação. Uma GMM foi usada como classificador. A base de dados usada é a ASVspoof 2017. Quanto aos resultados se obteve uma EER geral menor que 0,1.

A proposta de [Han18] é usar sinais residuais de predição linear, para, juntamente com coeficientes cepstrais criar características que serão apresentas a um classificador GMM. Novamente, a base de dados usada foi a ASVspoof 2015 e os resultados em ERR geral foram de 5,249.

Para detecção de voice spoofing [RBABA19] importa do campo de processamento de imagens o conceito de textura, para o processamento de voz esse conceito é chamado de "texturas de voz". Padrões binários locais (LBP) e seu respectivo histograma são usados para a construção do vetor de características que será avaliado por uma SVM. A base de dados usada para testes foi a ASVspoof 2015. A taxa máxima de acurácia conseguida foi de 0,7167.

Uma abordagem que combina análise de sinal de fala usando a *transformada de constante Q* (CQT) com o processamento cepstral é mostrada em [TDE17], essa técnica resulta no que se chama *coeficientes cepstrais de constante Q*(CQCCs), segundo o artigo, a vantagem destes coeficientes é a resolução espectro temporal variável. As base de dados usadas foram a RedDots [Pro], ASVspoof 2015 e AVSpoof. Foram usados três classificadores:

- DA-IICT: Uma fusão de dois classificadores GMM, sendo que um deles usa *coeficientes cepstrais de frequência MEL*(MFCC) e o outro usa características CFCC-IF [PP15].
- STC [NKL+16].
- SJTU [KMM+16].

Na seção de experimentos são feitos testes para cada uma das bases com os seguintes

resultados: ASVspoof 2015 → EER geral de 0.026; AVspoof → EER geral de 0; RedDots → EER geral de 0,185.

O artigo [SPM18] propõe uma aproximação usando reverberação e as partes não vozeadas da fala, três GMMs foram definidos para a classificação, esses classificadores votam se uma ocorrência é ou não verdadeira, ganhado sempre a classificação que obtiver mais votos. A base de dado utilizada foi a ASVSpoof 2017. O sistema de avaliação de desempenho escolhido, novamente, foi o ERR e esta alcançou um valor de 2,99.

A principal ideia de [KP18] é capturar a amplitude instantânea vinda de flutuações instantâneas de energia. Segundo o artigo as modulações de amplitude são mais suscetíveis ao ruído inserido no sinal original por uma fonte reprodutora. O estudo usa a base de dados ASVSpoof 2017 e GMM como classificador. Os resultados apresentados chegaram a uma EER de 0.0019.

No trabalho [WIAE18] foram usadas as diferenças entre bandas de frequências específicas para diferenciar um sinal legítimo de um usado em ataques de falsificação. Neste trabalho é proposta a *predição linear em domínio de frequência*(FDLP) juntamente com GMMs para classificação dos dados presentes na base ASVspoof 2017. Os resultados apresentados chegaram a uma EER de 0.0803.

Em [SSWA18] se propõe duas novas características que visam interpretar as componentes estáticas e dinâmicas do sinal, essas características complementam as características de tempo restrito no espectro. São elas a *"modulation spectral centroid frequency"* e a *long term spectral average*. O sistema usa como classificador um GMM juntamente com a base dados ASVSpoof 2017. Os resultados chegaram a um valor de EER de 0,0654.

Considerando o envelopamento das amplitudes e das frequências instantâneas em cada banda estreita filtrada, [KP17] discute como diferenciar um sinal legítimo de um falso. A base de dados usada foi a ASVSpoof 2015. Um GMM foi usado como classificador e, em relação ao desempenho, o método chegou a ter um EER de 0,045.

Neste trabalho [DGK+16] é proposto o uso do *gammatone frequency cepstral coefficients*(MGFCC). O gammatone é o produto de uma distribuição gamma com um sinal senoide e é usado na construção de filtros auditivos que, neste caso, são usados para

extrair características do sinal de voz. A base de dados usada foi a ASVspoof 2015. O classificador usado foi um GMM e o EER chegou a 0,02556.

Segundo [AKY+18] *Hashing* sensível a locus(LSH) é frequentemente usado como um classificador para problemas relacionados a *big data*, neste trabalho é proposto uma junção de MFCC e LSH a fim de se reconhecer o locutor. Neste método o MFCC é extraído dos arquivos de sinal para posterior aplicação do LSH gerando assim uma tabela *hash*, estes valores de *hash* são então comparados identificando assim o locutor ou locutora. Nos testes realizados houve uma acurácia de 92,66%. A base de dados usada foi a TIMIT 2018 [Con18].

Capítulo 3

Abordagem Proposta

3.1 Coleta de dados

Para a realização desta pesquisa coletou-se uma série de vozes nos arredores do Instituto de Biociências, Letras e Ciências Exatas em São José do Rio Preto no estado de São Paulo. Foram coletadas amostras de 21 Indivíduos, das quais 20 foram usadas já que, em um dos casos, não foi possível coletar todos os dados necessários. Tais gravações se constituem de dígitos em um intervalo de 0 a 9 falados tanto em língua inglesa como em portuguesa. Os locutores foram escolhidos de acordo com seu sexo e idade de forma que a amostra estudada tenha uma abrangência que cubra desde crianças em época pré-escolar até adultos entre 50 e 60 anos do sexo masculino e feminino.

As gravações foram feitas em ambientes distintos com diferentes níveis de ruído ao fundo garantindo uma boa variabilidade de interferências que certamente irão impactar nos resultados dos classificadores que serão usados.

Coletadas as amostras, os dígitos das mesmas foram separados um-a-um usando uma ferramenta desenvolvida para esse fim, resultando em uma base de dados com 820 trechos. No apêndice deste documento é possível entender a ferramenta criada para auxiliar na preparação desta base de dados.

3.2 Formato dos dados

Foram usados arquivos no formato *wave pulse-code modulation* (PCM), neste esquema, fora os descartes de informação que ocorrem pela da discretização do sinal. Os dados são armazenados sem perdas significativas para a análise necessária, já que não são empregados métodos de corte nas frequências ou outro método de compactação destrutiva.

A taxa de amostragem escolhida é 44100hz, que permite, segundo o teorema de Nyquist, que seja realizada a quantização de frequências de até 22050hz. A resolução dos valores é 16bits.

3.2.1 Áudios originais

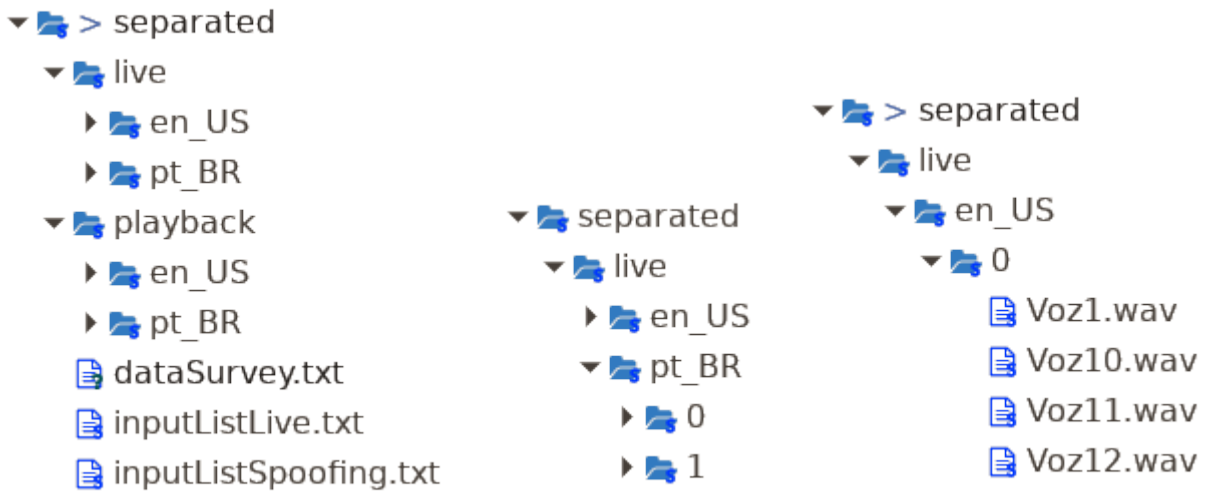
Para a constituição da base de dados não regravada os áudios originais foram editados e separados digito a dígito.

3.2.2 Áudios regravados

No caso da base usada para simulação de *voice spoofing* foi criado um arquivo contendo todas as falas de todos os entrevistados, em seguida, os sons reproduzidos por este foram regravados por um segundo dispositivo de gravação diferente do original.

3.3 Organização da base de dados

A organização da base de dados se deu por tipo (regravado ou não), idioma, dígito ditado e interlocutor considerado. Foi criada uma estrutura hierárquica de diretórios de forma a permitir que fosse fácil e intuitivo acessar cada uma das amostras seja por vias automatizadas ou não. Os arquivos regravados residem no diretório "playback" já os não regravados se encontram em "live". Essa organização é mostrada nas figuras 3.1(a), 3.1(b) e 3.1(c).



(a) Base em nível 1 (b) Base em nível 2 (c) Base em nível 3

Figura 3.1: Organização da base de dados

Para facilitar a automação do processamento foram criados três arquivos de texto:

- ***dataSurvey.txt***: Contêm os dados de idade e sexo de cada entrevistado.
- ***inputListLive.txt***: Uma lista de caminhos para todos os arquivo não regravados.
- ***inputListSpoofing.txt***: Apresenta uma listagem dos caminhos para todos os arquivos regravados.

Apenas para ilustrar, o conteúdo do diretório **"separated/live/en_US/0"** se constitui de vários arquivos do tipo *wave* cada um identificando o locutor ao qual pertence como mostrado na figura 3.1(c).

3.4 Wavelets

Como já explanado no capítulo 2, neste trabalho apenas as transformadas diretas serão realizadas já que, neste contexto, não interessa a reconstrução do sinal. A abordagem usada será baseada nos filtros de análise digitais que proporcionarão a decomposição do sinal com o uso de filtros passa baixas e passa altas estritamente no domínio discreto.

3.4.1 Wavelets testadas

Nós experimentos explicados mais a frente foram testadas as seguintes wavelets:

- haar
- daub4
- daub6
- daub8
- daub10
- daub12
- daub14
- daub16
- daub18
- daub20
- daub22
- daub24
- daub26
- daub28
- daub30
- daub32
- daub34
- daub36
- daub38
- daub40
- daub42
- daub44
- daub46
- daub48
- daub50
- daub52
- daub54
- daub56
- daub58
- daub60
- daub62
- daub64
- daub66
- daub68
- daub70
- daub72
- daub74
- daub76
- sym8
- sym16
- sym32
- coif6
- coif12
- coif18
- coif24
- coif30
- beylkin18
- vaidyanathan24

3.5 Experimentos

3.5.1 Experimento 01 - Melhor combinação Wavelet x BARK ou MEL

Apresentação

O objetivo desse experimento verificar, segundo análise paraconsistente, qual das combinações entre as escalas BARK ou MEL e as várias *wavelets* consideradas geram os vetores de características com menos sobreposição (disjuntos).

Mais informações sobre a análise paraconsistente se encontram no capítulo 2.

O uso de *wavelets* geralmente tem como objetivo a geração de vetores que serão analisados ou processados posteriormente, tais sequências de valores carregam a infor-

mação de frequência e tempo que podem ser úteis para uma análise específica. Neste trabalho, porém, a decomposição deve ser feita até seu limite, pois, após isso será realizada a soma das energias nos intervalos definidos pela escala BARK e MEL que seriam impossíveis, ou imprecisas se assim não fosse. Por fim, é feito o posicionamento dos resultados no plano paraconsistente.

O algoritmo para a realização das etapas é mostrado na listagem 3.1.

Redimensionando o sinal

Antes da aplicação da transformada *wavelet* é necessário redimensionar o sinal para que este tenha um comprimento correspondente a uma potência de 2 como indicado na equação 3.1. Isso é necessário para que não haja perdas de informação ao final da transformação, pois, como a transformada *wavelet* realiza o *downsampling* numa ordem de 2, ou seja, em cada nível o tamanho do vetor do sinal é diminuído pela metade. Caso haja um comprimento diferente do citado, em alguma parte do processo a divisão não será inteira fazendo com que algum/alguns valores dentro do processamento não sejam considerados.

Na equação 3.1 ***proxInt*** é uma função que retorna o próximo número inteiro dado outro real (por exemplo, $proxInt(1,5) = 2$) e ***tamanho*** é a quantidade de itens no vetor do sinal.

$$tamanhoOtimo = 2^{proxInt(\log_2 tamanho)} \tag{3.1}$$

Determinando o número máximo de transformações

Após o redimensionamento do sinal a quantidade máxima de transformações necessárias para reduzir os coeficientes a um vetor unitário é dada pela equação 3.2.

$$maxTrans = \log_2 tamanho \tag{3.2}$$

Algoritmo

```
// Carregue para a memoria um dos conjuntos de amostra
for (listaDeAmostras : {listaComVoiceSpoofing, listaSemVoiceSpoofing}) {
  // Selecione o proximo tipo de wavelet
  for (wavelet : wavelets) {
    // Selecione entre BARK ou MEL
    for (barkOuMel : {BARK, MEL}) {
      // Selecione o proximo sinal dentro da amostra
      for (sinal : listaDeAmostras) {
        tamanhoOtimo=calcularTamanhoOtimo(sinal);
        redimensionar(sinal, tamanhoOtimo);
        sinalTransformado=wavelet(sinal, wavelet);
        energias=calcularEnergias(sinalTransformado, barkOuMel);
        energias=normalizar(energias);

        // Armazene os resultados
        resultados[wavelet.nome()][barkOuMel][listaDeAmostras.nome()].adicionar(
    energias);
      }
    }
  }
}
// Posicione os resultados no plano paraconsistente
mostraResultadosNoPlanoParaconsistente(resultados);
```

Listing 3.1: Algoritmo do experimento 1

3.5.2 Experimento 02 - Desempenho da combinação mais disjunta em classificadores por distância

Apresentação

O objetivo deste é verificar, considerando as melhores combinações descobertas pelo experimento 1, o desempenho de classificadores por distâncias Euclidiana e Manhattan. Nesta fase do estudo os vetores de características gerados pelo experimento 1 são fornecidos aos classificadores cujos desempenhos serão medidos.

Objetivando medir o comportamento da classificação com diversos tamanhos para os modelos de referência, o tamanho dos mesmos foi definido segundo porcentagens da quantidade total de amostras.

Para cada porcentagem escolhida o experimento deve ser uma vez, depois duas vezes, após isso 3, 4, 5, até que se chegue a execução de um número definido de testes dado pela equação 3.3. Em cada uma dessas execuções, a ordem dos sinais dentro das amostras (*spoofing* e não *spoofing*) foi permutada aleatoriamente para evitar que uma coincidência afetasse os valores dos resultados.

Abaixo: ***numTestes*** representa número máximo de testes que podem ser feitos usando uma certa porcentagem.

$$qtdeTotalDeTestes = numTestes * \frac{numTestes + 1}{2} \tag{3.3}$$

Para cada porcentagem foram coletadas as melhores e as piores acurácias assim como suas respectivas matrizes de confusão.

O algoritmo deste experimento é mostrado na listagem 3.2.

Amostragem

Houve uma separação entre amostras para teste e outras constituir o modelo de referência para o classificador. Os modelos de referência foram dimensionados usando-se 10%, 20%, 30%, 40% e 50% da quantidade total de amostras para ambas as classes (*spoofing* e não *spoofing*) as restantes foram usadas nos testes.

Algoritmo

```
1 tamanhosDoModelo={0.1, 0,2, 0,4, 0,5};
2 modeloDeReferenciaNaoSpoofing={};
3 modeloDeReferenciaSpoofing={};
4 testesNaoSpoofing={};
5 testesSpoofing={};
6
```

```
for (distancia : {Euclidiana, Manhattan}) {
  for(porcentagem : tamanhosDoModelo){
    for(teste = 0; teste < 300; teste++){
      // Escolhe aleatoriamente os sinais para o modelo com spoofing
      // e os grava em 'modeloDeReferenciaSpoofing' o restante vai
      // para 'testesSpoofing'
      escolherAleatoriamente(listaComVoiceSpoofing, porcentagem,
    modeloDeReferenciaSpoofing, testesSpoofing);

      // Escolhe aleatoriamente os sinais para o modelo sem spoofing
      // e os grava em 'modeloDeReferenciaNaoSpoofing' o restante vai
      // para 'testesNaoSpoofing'
      escolherAleatoriamente(listaSemVoiceSpoofing, porcentagem,
    modeloDeReferenciaNaoSpoofing, testesNaoSpoofing);
      treinarClassificador("spoofing", modeloDeReferenciaSpoofing);
      treinarClassificador("naoSpoofing", modeloDeReferenciaNaoSpoofing);

      // Classifica os testes e preenche a tabela de confusao
      for(sinal : testesSpoofing){
        preencherTabelaDeConfusao(sinal, "spoofing");
      }

      // Classifica os testes e preenche a tabela de confusao
      for(sinal : testesNaoSpoofing){
        preencherTabelaDeConfusao(sinal, "naoSpoofing");
      }

      acuracia=calculaAcuracia();

      // Salva a melhor acuracia e matriz de confusao
      if(ehAMelhorAcuracia(acuracia)){
        salvaAcuraciaEMatrizDeConfusao();
      }

      // Salva a pior acuracia e matriz de confusao
```

```
40          if(ehAPiorAcuracia(acuracia)){
41            salvaAcuraciaEMatrizDeConfusao();
42          }
43        }
44    }
45 }
```

Listing 3.2: Algoritmo do experimento 2

3.5.3 Experimento 03 - Desempenho da combinação mais disjunta em uma SVM.

Apresentação

Considerando as melhores combinações descobertas pelo experimento 1, nesta parte pretende-se medir o desempenho de uma *Support Vector Machine* (SVM). Nesta fase do estudo os vetores de características gerados pelo experimento 1 são fornecidos ao classificador cujo desempenho será medido. São idênticas ao experimento 2 as quantidades de testes, as formas de separação dos conjuntos de modelo e teste (amostragem), assim como boa porção do algoritmo, este, será apresentado apenas como garantia do entendimento e se encontra na listagem 3.3.

Algoritmo

```
1 tamanhosDoModelo={0.1, 0,2, 0,4, 0,5};
2 modeloDeReferenciaNaoSpoofing={};
3 modeloDeReferenciaSpoofing={};
4 testesNaoSpoofing={};
5 testesSpoofing={};
6   for(porcentagem : tamanhosDoModelo){
7     for(teste = 0; teste < 300; teste++){
8       // Escolhe aleatoriamente os sinais para o modelo com spoofing
9       // e os grava em 'modeloDeReferenciaSpoofing' o restante vai
10      // para 'testesSpoofing'
```

```
11      escolherAleatoriamente(listaComVoiceSpoofing, porcentagem,
    modeloDeReferenciaSpoofing, testesSpoofing);
12
13      // Escolhe aleatoriamente os sinais para o modelo sem spoofing
14      // e os grava em 'modeloDeReferenciaNaoSpoofing' o restante vai
15      // para 'testesNaoSpoofing'
16      escolherAleatoriamente(listaSemVoiceSpoofing, porcentagem,
    modeloDeReferenciaNaoSpoofing, testesNaoSpoofing);
17      treinarClassificador("spoofing", modeloDeReferenciaSpoofing);
18      treinarClassificador("naoSpoofing", modeloDeReferenciaNaoSpoofing);
19
20      // Classifica os testes e preenche a tabela de confusao
21      for(sinal : testesSpoofing){
22        preencherTabelaDeConfusao(sinal, "spoofing");
23      }
24
25      // Classifica os testes e preenche a tabela de confusao
26      for(sinal : testesNaoSpoofing){
27        preencherTabelaDeConfusao(sinal, "naoSpoofing");
28      }
29
30      acuracia=calculaAcuracia();
31
32      // Salva a melhor acuracia e matriz de confusao
33      if(ehAMelhorAcuracia(acuracia)){
34        salvaAcuraciaEMatrizDeConfusao();
35      }
36
37      // Salva a pior acuracia e matriz de confusao
38      if(ehAPiorAcuracia(acuracia)){
39        salvaAcuraciaEMatrizDeConfusao();
40      }
41    }
42  }
```

Listing 3.3: Algoritmo do experimento 3

Capítulo 4

Testes e Resultados

4.1 Experimento 01

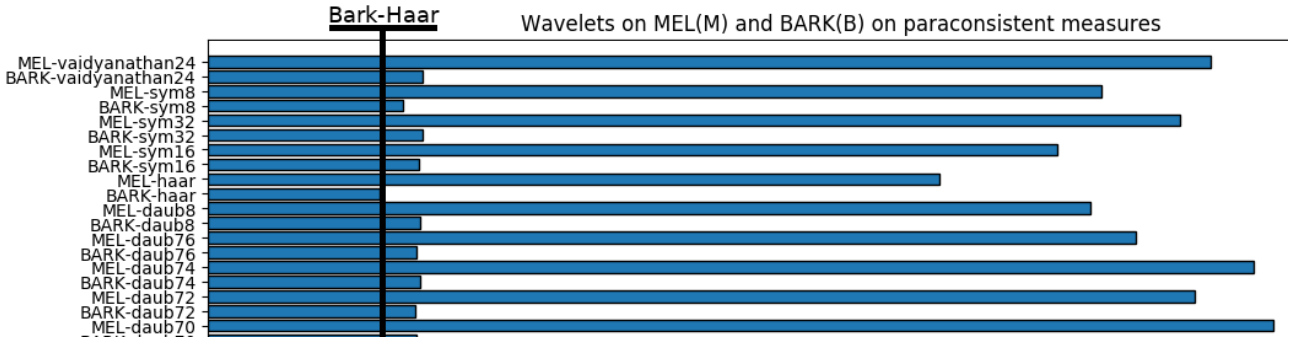


Figura 4.1: Distância da combinação Wavelet*X*BARK ou MEL do ponto (1,0)

Wavelet	G1	G2	Distancia do ponto (1,0)
haar	0.93615	4.68316e-310	0.0638503
daub4	0.928088	4.68316e-310	0.0719123
daub6	0.927885	4.68316e-310	0.072115
coif6	0.927823	4.68316e-310	0.072177
sym8	0.92769	4.68316e-310	0.0723096
daub12	0.926541	4.68316e-310	0.073459

Tabela 4.1: Wavelet*X*BARK no plano paraconsistente

Na figura 4.1 quanto menor o valor, mais disjuntos tendem ser os vetores de características gerados para as duas classes testadas (*spoofing* e não *spoofing*). Como se pode

constatar, das combinações testadas, a ***Haar com BARK*** conseguiu o **melhor desempenho** na criação dos melhores vetores de características. A tabela 4.1 mostra os 6 melhores resultados em distância do ponto (1,0)(verdade) no plano paraconsistente, a totalidade dos dados se encontram no apêndice deste documento nas tabelas 6.1 e 6.2 para todos os resultados com BARK e nas tabelas 6.3 e 6.4 para os com MEL. Na figura 6.2 se pode ver o gráfico de barras para todas as combinações *wavelets*X*BARK/MEL*.

A combinação ***wavelet + BARK*** teve, consistentemente, um desempenho **melhor** do que as respectivas combinações ***wavelet + MEL***.

4.2 Experimento 02

Considerando que o experimento 1 teve como melhor resultado a combinação **Haar+BARK** o objetivo deste é constatar a máxima acurácia se consegue atingir em um classificador baseado em distâncias Euclidianas e Manhattan. O dimensionamento do tamanho das amostras de referência que neste caso são os itens usados para o treinamento do classificador variou em 10%, 20%, 30%, 40% e finalmente 50% do total das amostras. 300 foi a quantidade máxima de testes escolhida.

Os resultados gerais são mostrados na tabela **??**.

Mais níveis de detalhes para a distância euclidiana pode ser conseguido consultando-se as tabelas 4.3, 4.4, 4.5, 4.6, 4.7 e seus respectivos gráficos 4.2, 4.3, 4.4, 4.5, 4.6.

E para a distância Manhattan podem ser consultas as tabelas 4.8, 4.9, 4.10, 4.11, 4.12 e seus respectivos gráficos 4.7, 4.8, 4.9, 4.10, 4.11.

Tamanho do modelo	Acurácia mínima	Acurácia máxima
10%	0,6666	0,8861
20%	0,7439	0,8902
30%	0,7665	0,8919
40%	0,7784	0,9024
50%	0,7804	0,9097

Tabela 4.2: Resultados do experimento 02

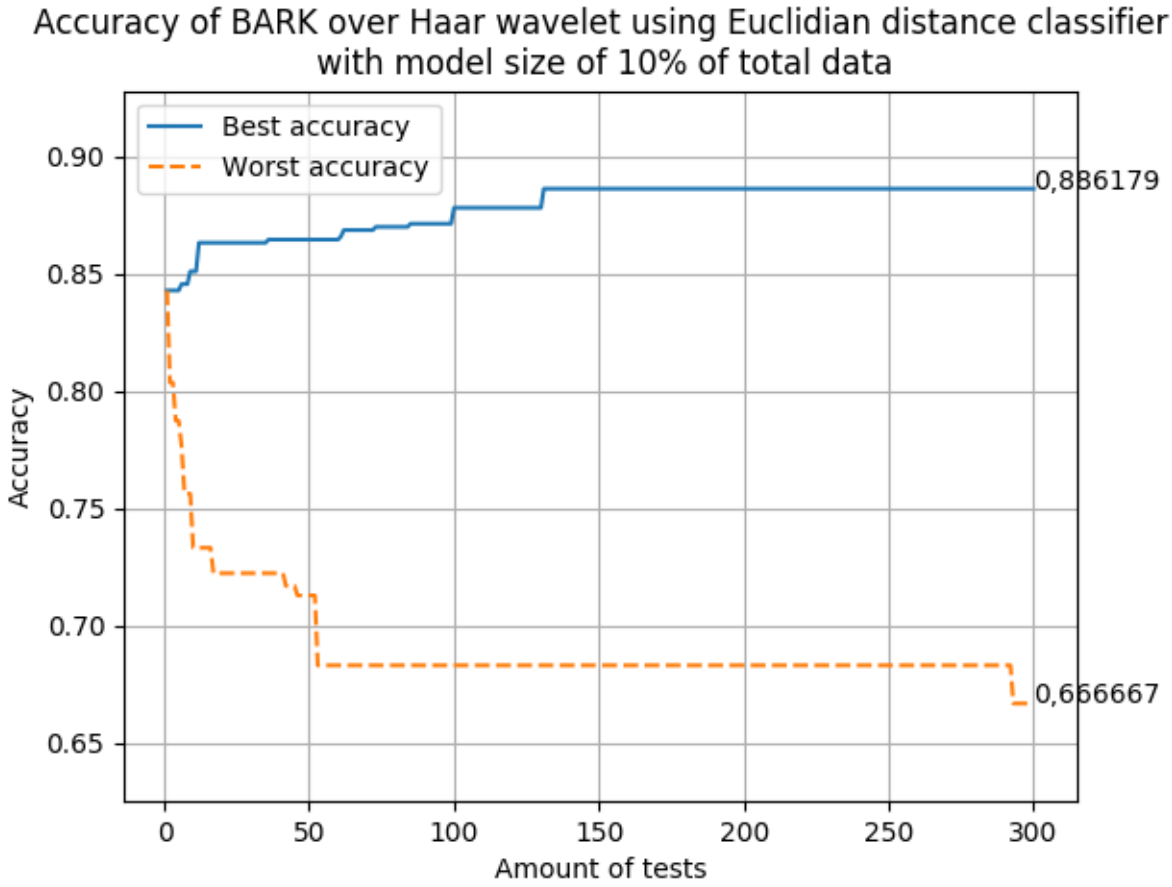


Figura 4.2: Acurácia *X* quantidade de testes - Distância Euclidiana, modelo a 10%

	Verdadeiro	**Falso**
Verdadeiro	339	54
Falso	30	315

Tabela 4.3: Tabela de confusão para classificador Euclidiano 10%

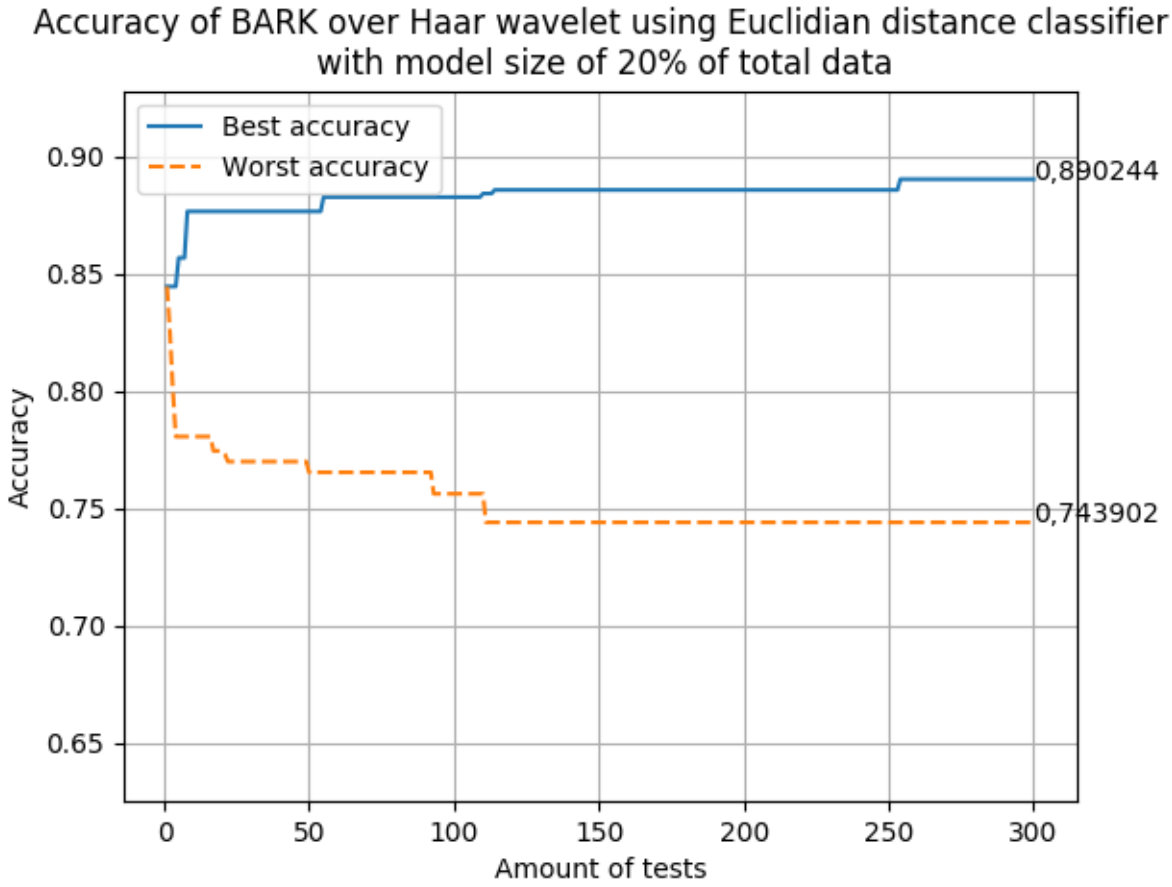


Figura 4.3: Acurácia *X* quantidade de testes - Distância Euclidiana, modelo a 20%

	Verdadeiro	**Falso**
Verdadeiro	310	54
Falso	18	274

Tabela 4.4: Tabela de confusão para classificador Euclidiano 20%

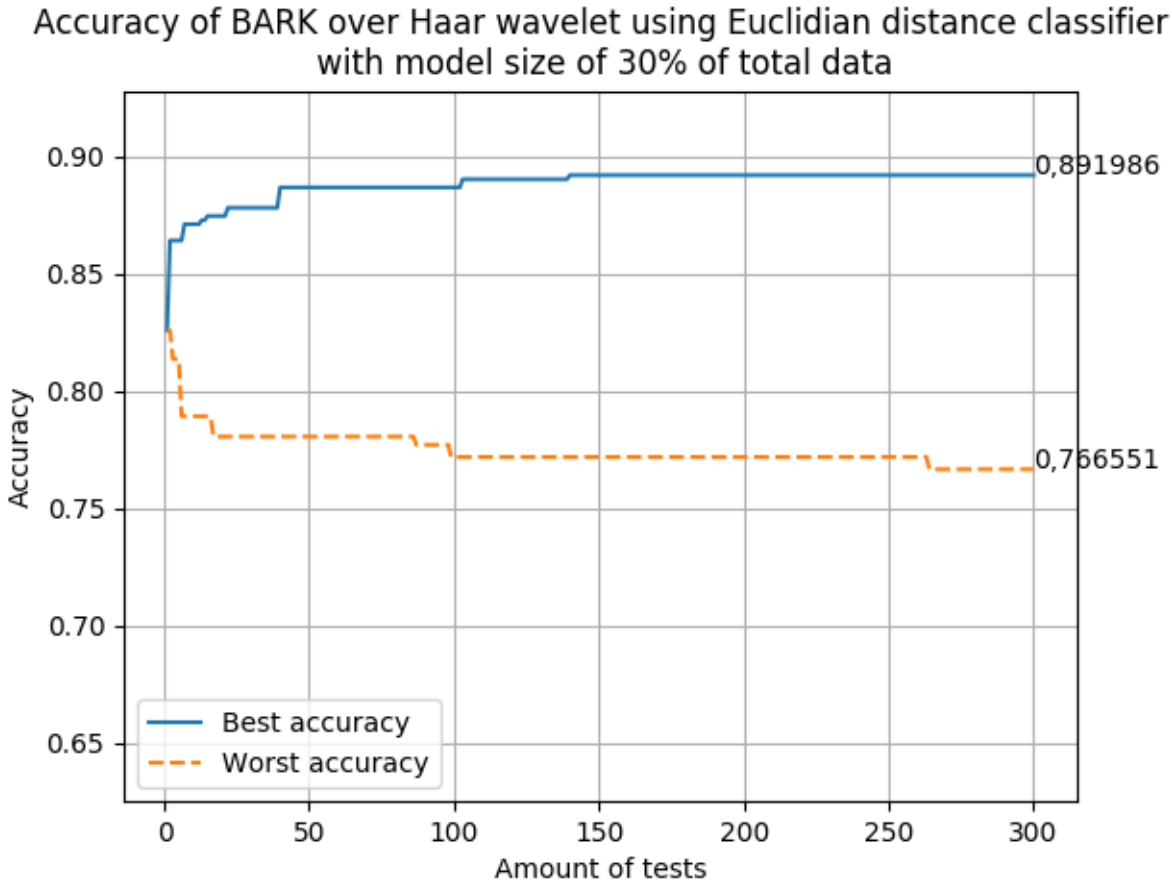


Figura 4.4: Acurácia *X* quantidade de testes - Distância Euclidiana, modelo a 30%

	Verdadeiro	**Falso**
Verdadeiro	263	38
Falso	24	249

Tabela 4.5: Tabela de confusão para classificador Euclidiano 30%

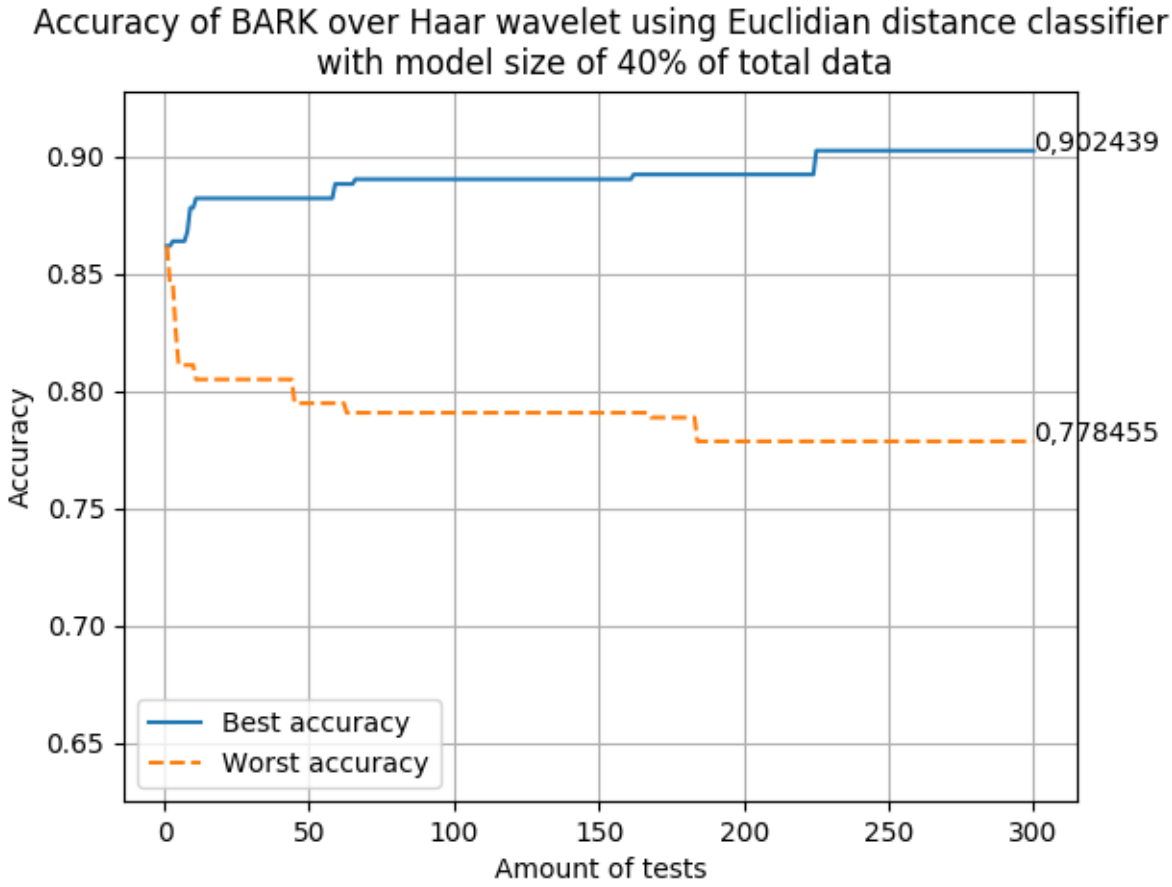


Figura 4.5: Acurácia *X* quantidade de testes - Distância Euclidiana, modelo a 40%

	Verdadeiro	**Falso**
Verdadeiro	233	35
Falso	13	211

Tabela 4.6: Tabela de confusão para classificador Euclidiano 40%

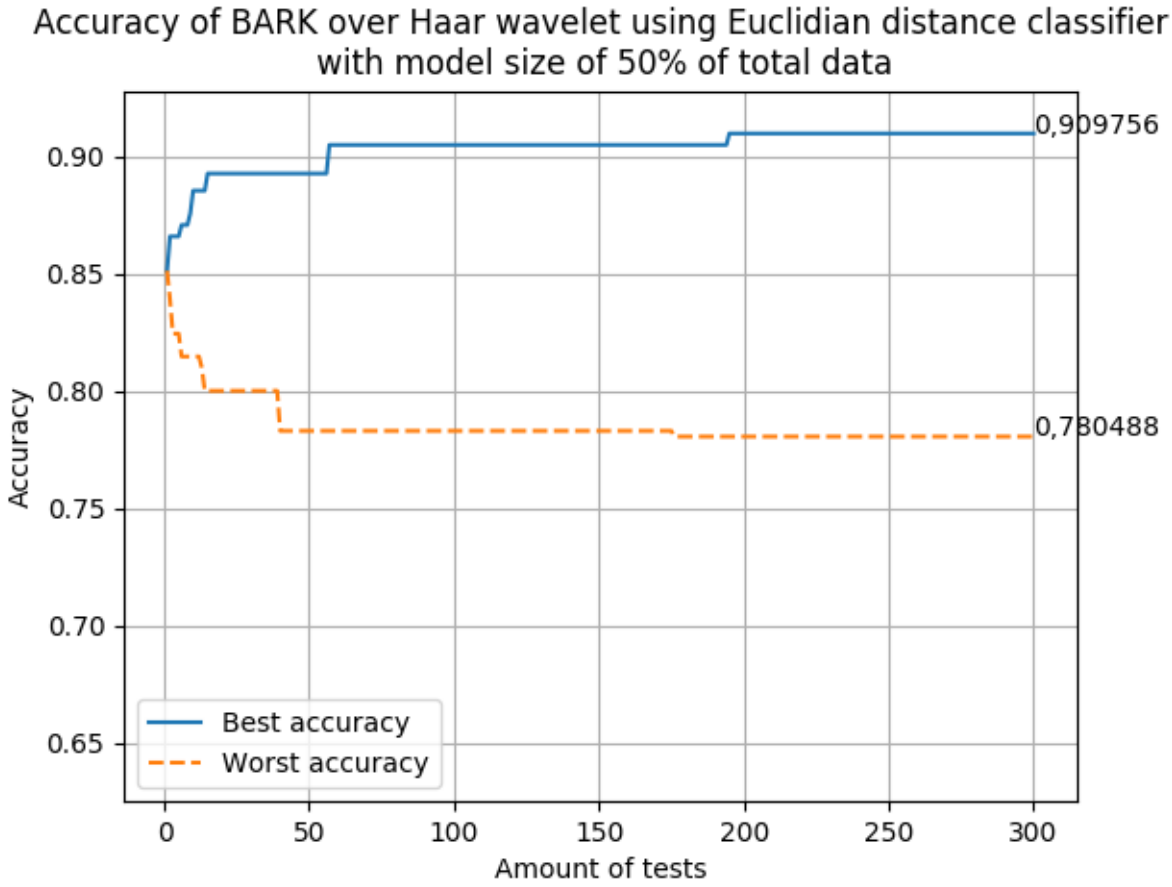


Figura 4.6: Acurácia *X* quantidade de testes - Distância Euclidiana, modelo a 50%

	Verdadeiro	**Falso**
Verdadeiro	198	30
Falso	7	175

Tabela 4.7: Tabela de confusão para classificador Euclidiano 50%

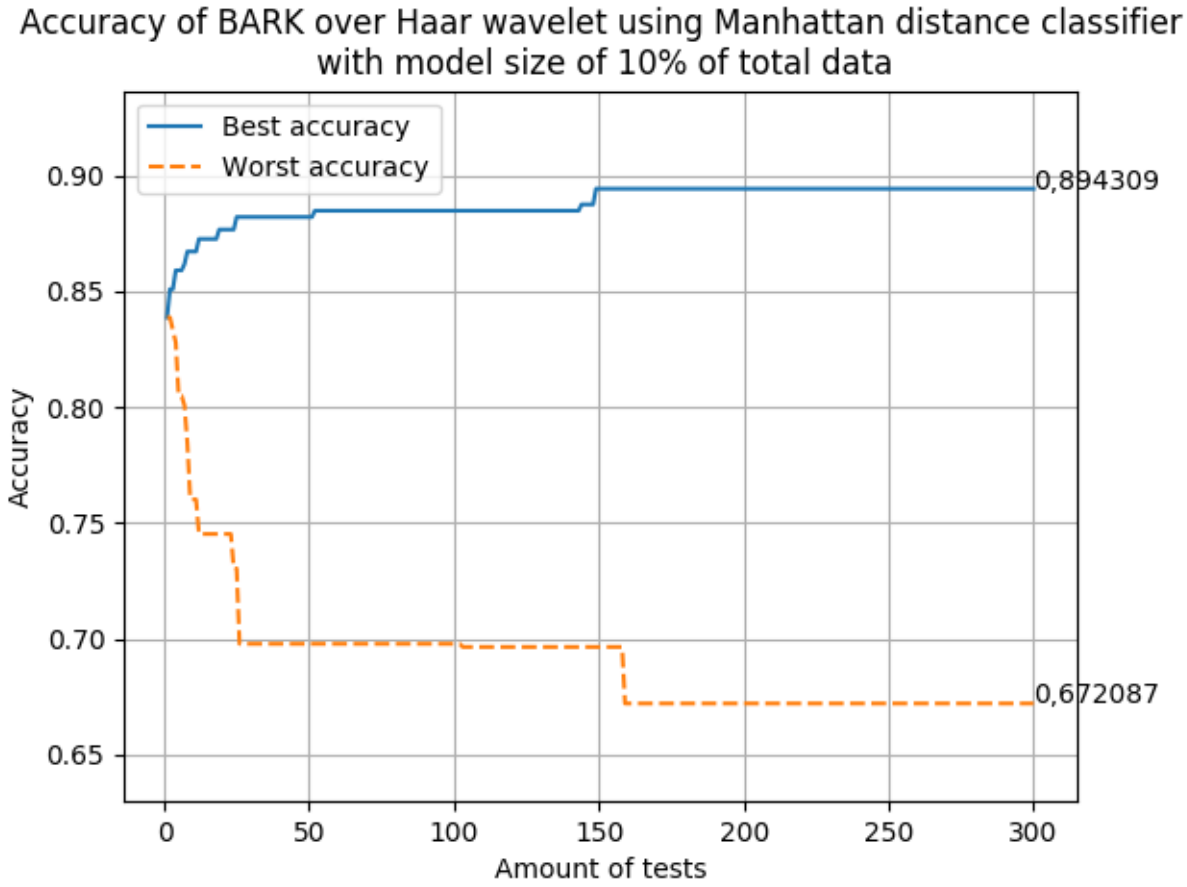


Figura 4.7: Acurácia *X* quantidade de testes - Distância Manhattan, modelo a 10%

	Verdadeiro	**Falso**
Verdadeiro	349	58
Falso	20	311

Tabela 4.8: Tabela de confusão para classificador Manhattan 10%

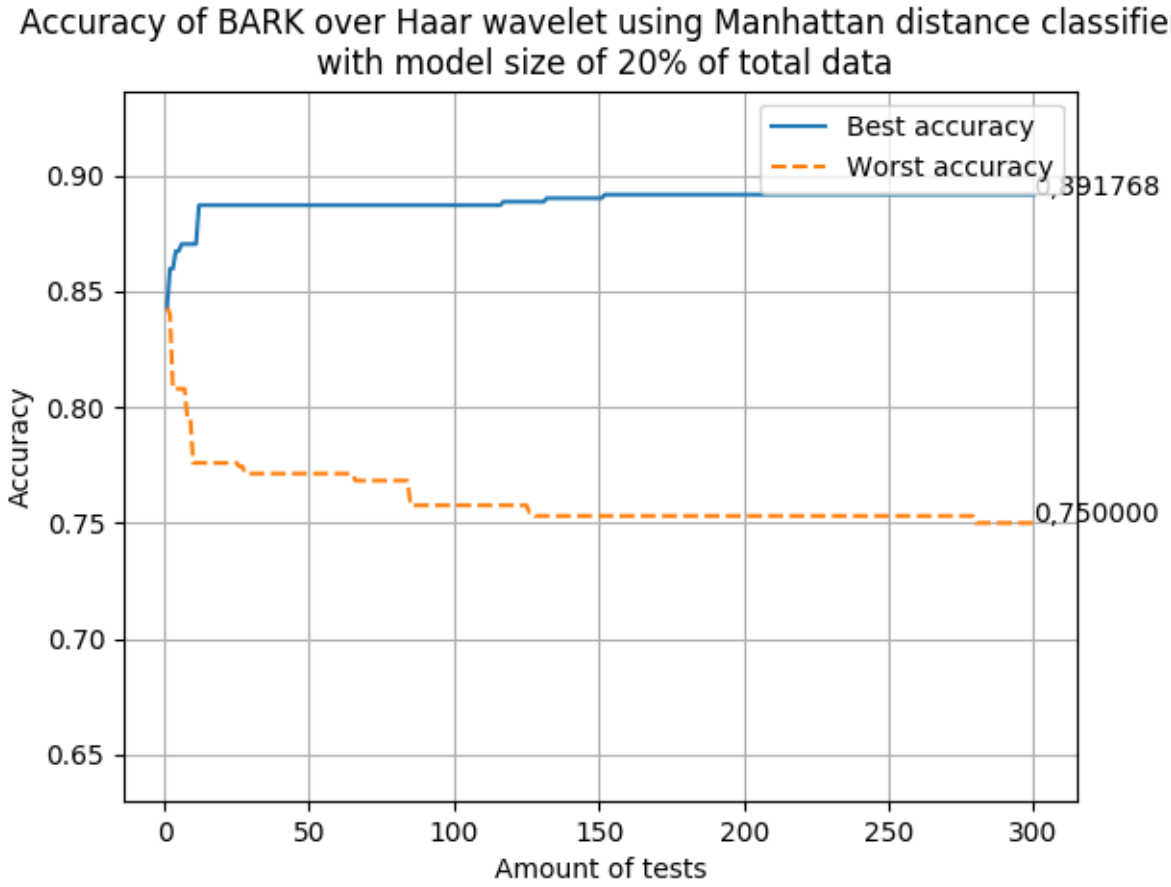


Figura 4.8: Acurácia *X* quantidade de testes - Distância Manhattan, modelo a 20%

	Verdadeiro	**Falso**
Verdadeiro	307	50
Falso	21	278

Tabela 4.9: Tabela de confusão para classificador Manhattan 20%

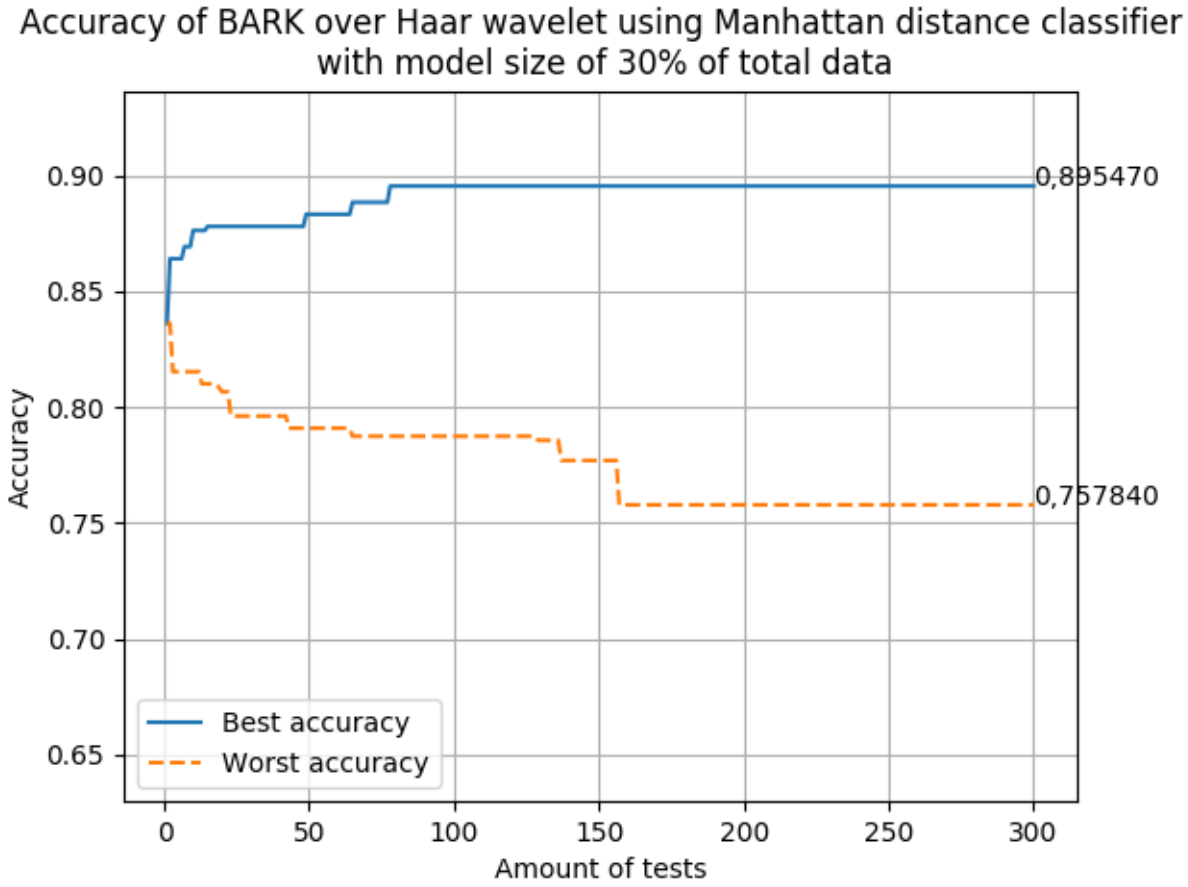


Figura 4.9: Acurácia *X* quantidade de testes - Distância Manhattan, modelo a 30%

	Verdadeiro	**Falso**
Verdadeiro	275	48
Falso	12	239

Tabela 4.10: Tabela de confusão para classificador Manhattan 30%

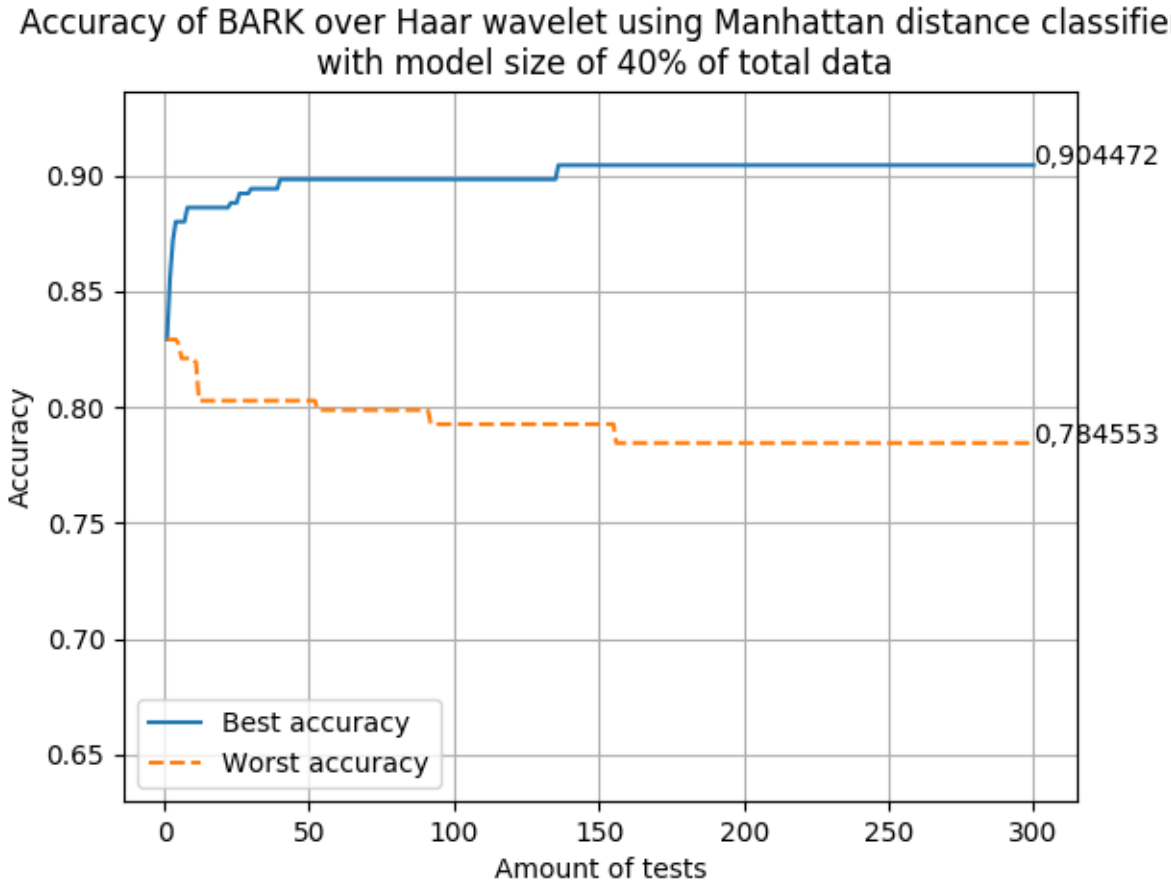


Figura 4.10: Acurácia *X* quantidade de testes - Distância Manhattan, modelo a 40%

	Verdadeiro	Falso
Verdadeiro	231	32
Falso	15	214

Tabela 4.11: Tabela de confusão para classificador Manhattan 40%

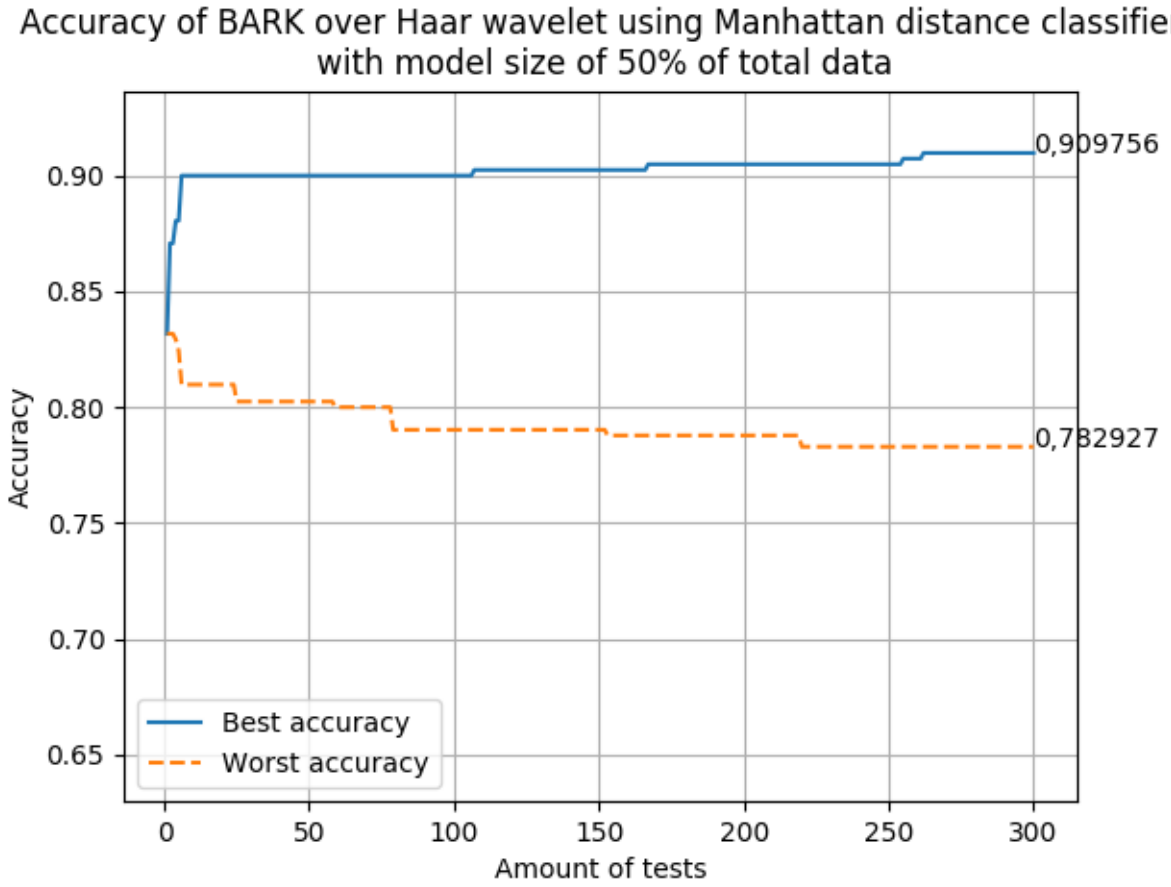


Figura 4.11: Acurácia *X* quantidade de testes - Distância Manhattan, modelo a 50%

	Verdadeiro	Falso
Verdadeiro	195	27
Falso	10	178

Tabela 4.12: Tabela de confusão para classificador Manhattan 50%

4.3 Experimento 03

Novamente, considerando que o experimento 1 teve como melhor resultado a combinação **Haar+BARK** o objetivo deste é constatar a máxima acurácia se consegue atingir em uma SVM. O dimensionamento do tamanho das amostras para o treinamento do classificador variou em 10%, 20%, 30%, 40% e finalmente 50% do total das amostras. 300 foi a quantidade máxima de testes escolhida.

Os resultados gerais são mostrados na tabela 4.13.

Mais níveis de detalhes podem ser consultados nas tabelas 4.14, 4.4, 4.16, 4.6, 4.18 e seus respectivos gráficos 4.12, 4.13, 4.14, 4.15, 4.16.

Tamanho do modelo	Acurácia mínima	Acurácia máxima
10%	0,8130	0,9200
20%	0,8506	0,9314
30%	0,8623	0,9390
40%	0,8638	0,9451
50%	0,8509	0,9512

Tabela 4.13: Resultados do experimento 03

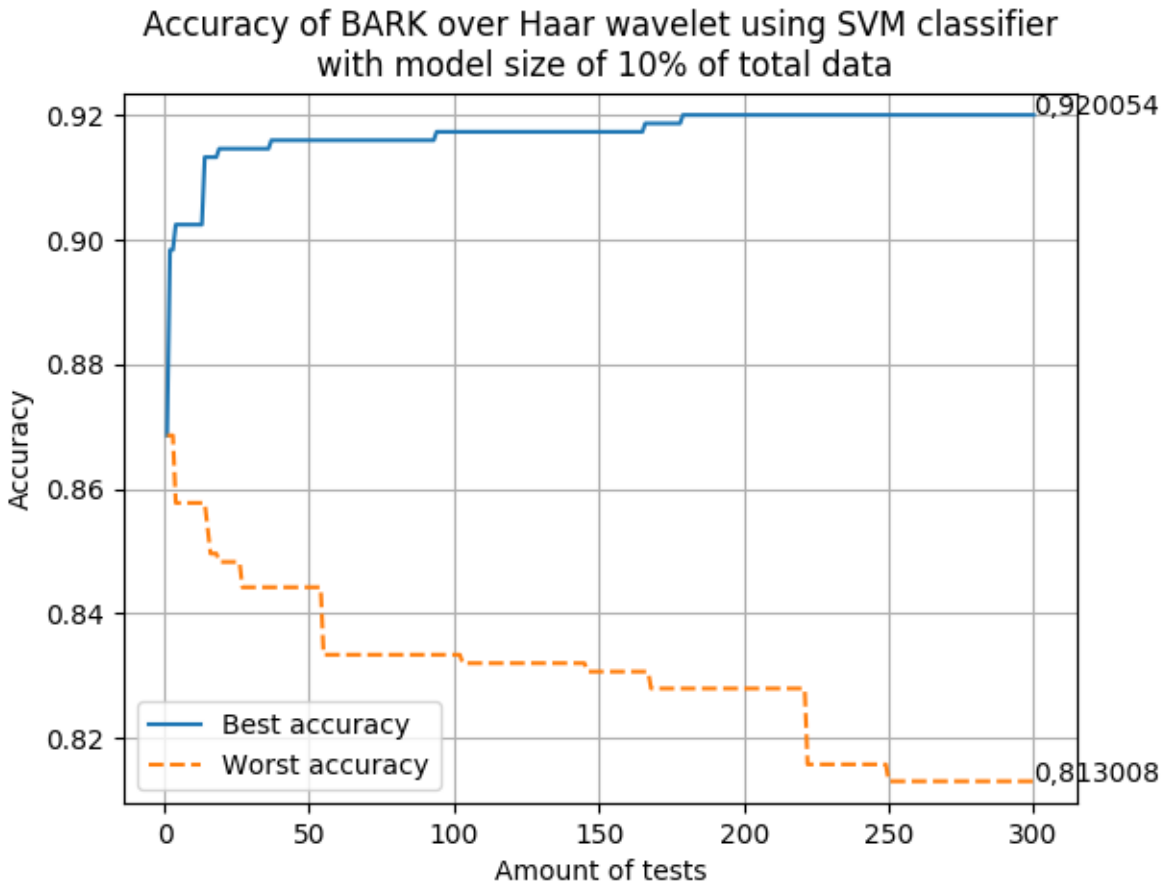


Figura 4.12: Acurácia *X* quantidade de testes - SVM, modelo a 10%

	Verdadeiro	**Falso**
Verdadeiro	331	21
Falso	38	348

Tabela 4.14: Tabela de confusão para classificador SVM 10%

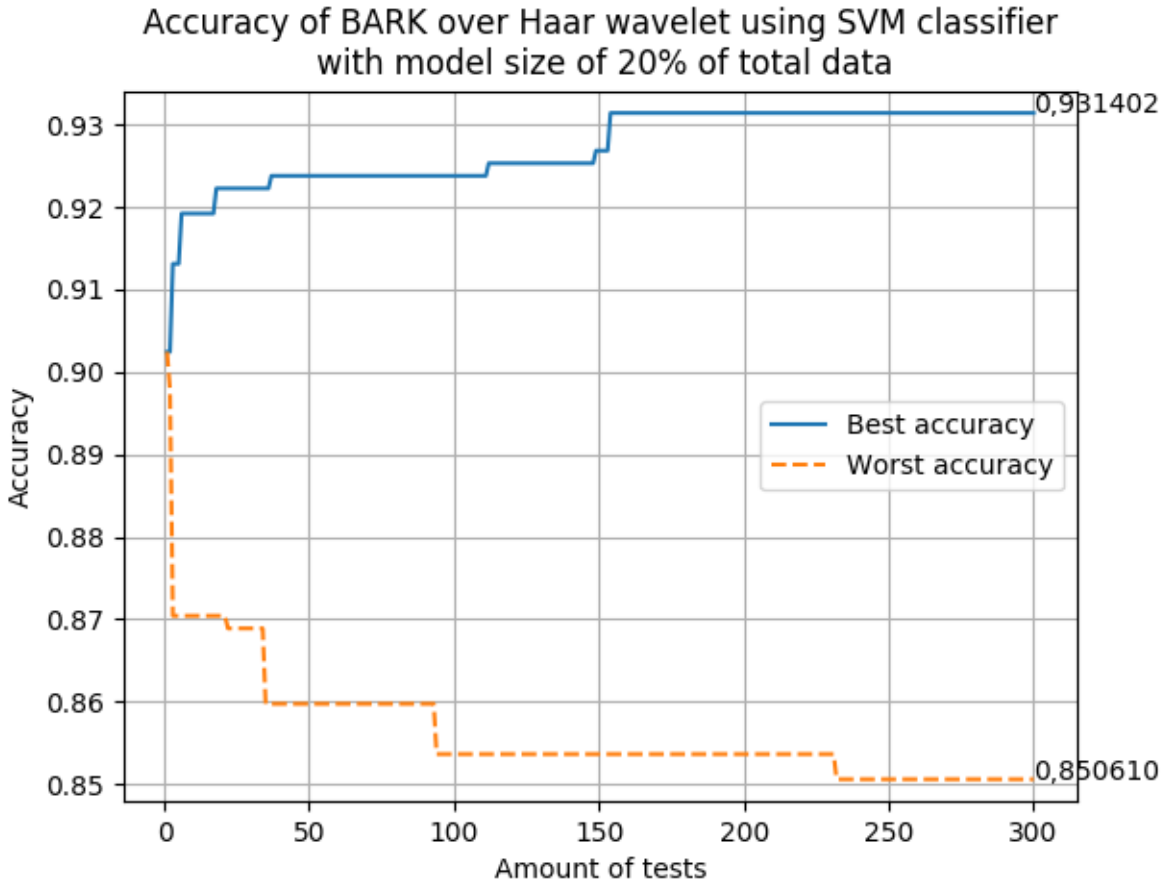


Figura 4.13: Acurácia *X* quantidade de testes - SVM, modelo a 20%

	Verdadeiro	**Falso**
Verdadeiro	301	18
Falso	27	310

Tabela 4.15: Tabela de confusão para classificador SVM 20%

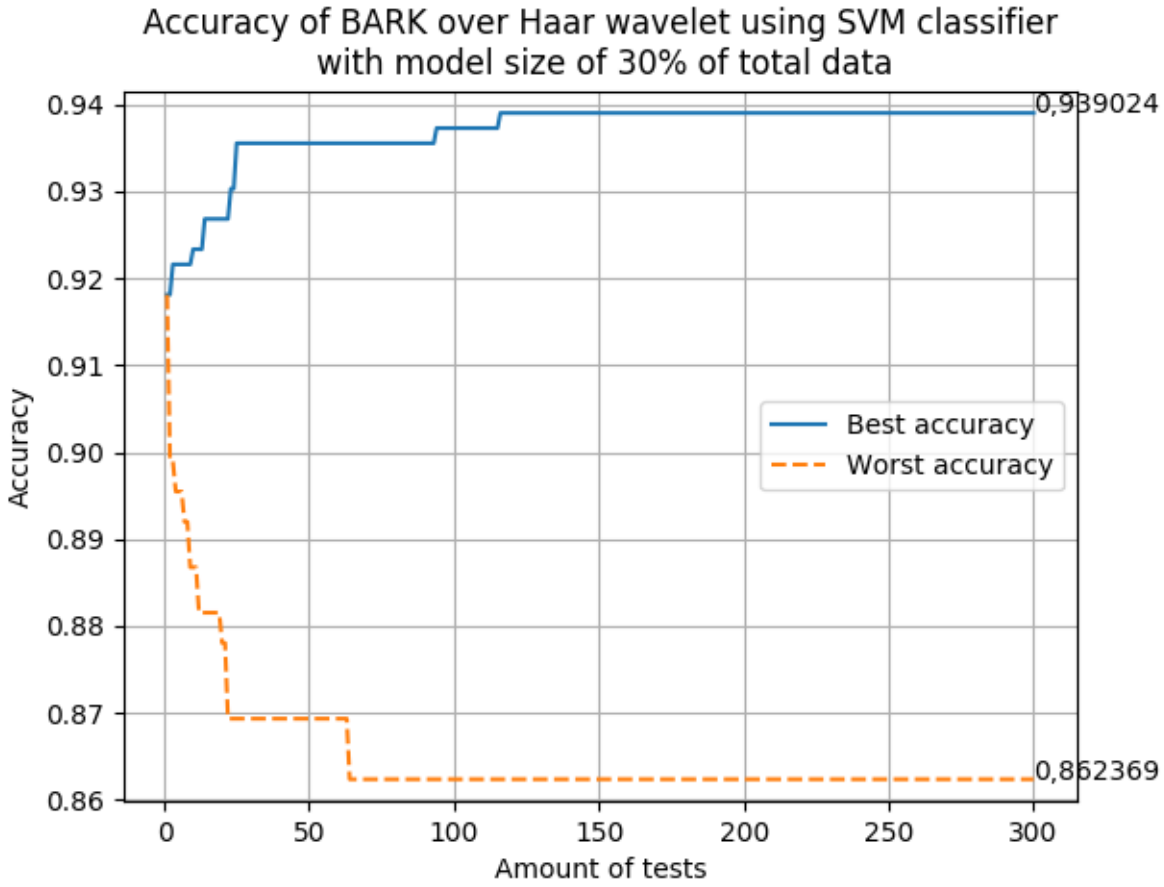


Figura 4.14: Acurácia *X* quantidade de testes - SVM, modelo a 30%

	Verdadeiro	**Falso**
Verdadeiro	269	17
Falso	18	270

Tabela 4.16: Tabela de confusão para classificador SVM 30%

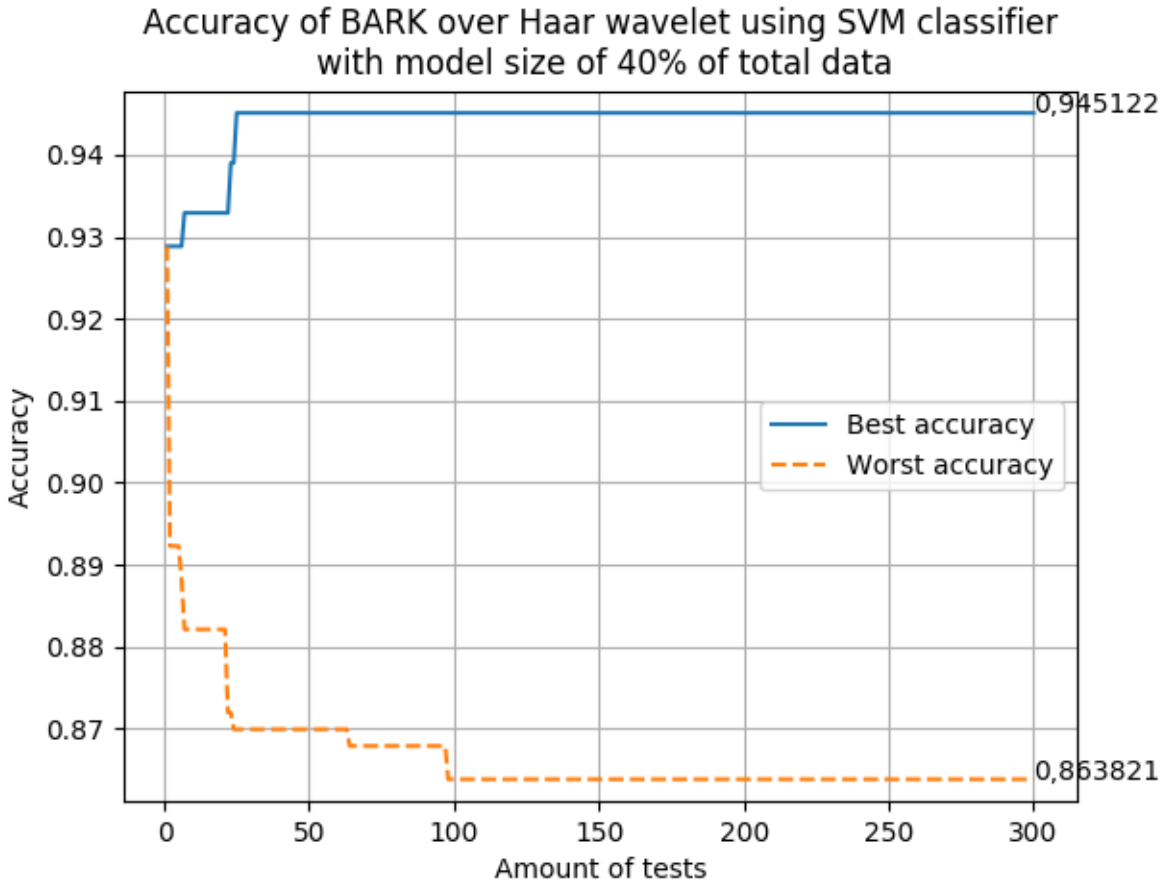


Figura 4.15: Acurácia *X* quantidade de testes - SVM, modelo a 40%

	Verdadeiro	**Falso**
Verdadeiro	235	16
Falso	11	230

Tabela 4.17: Tabela de confusão para classificador SVM 40%

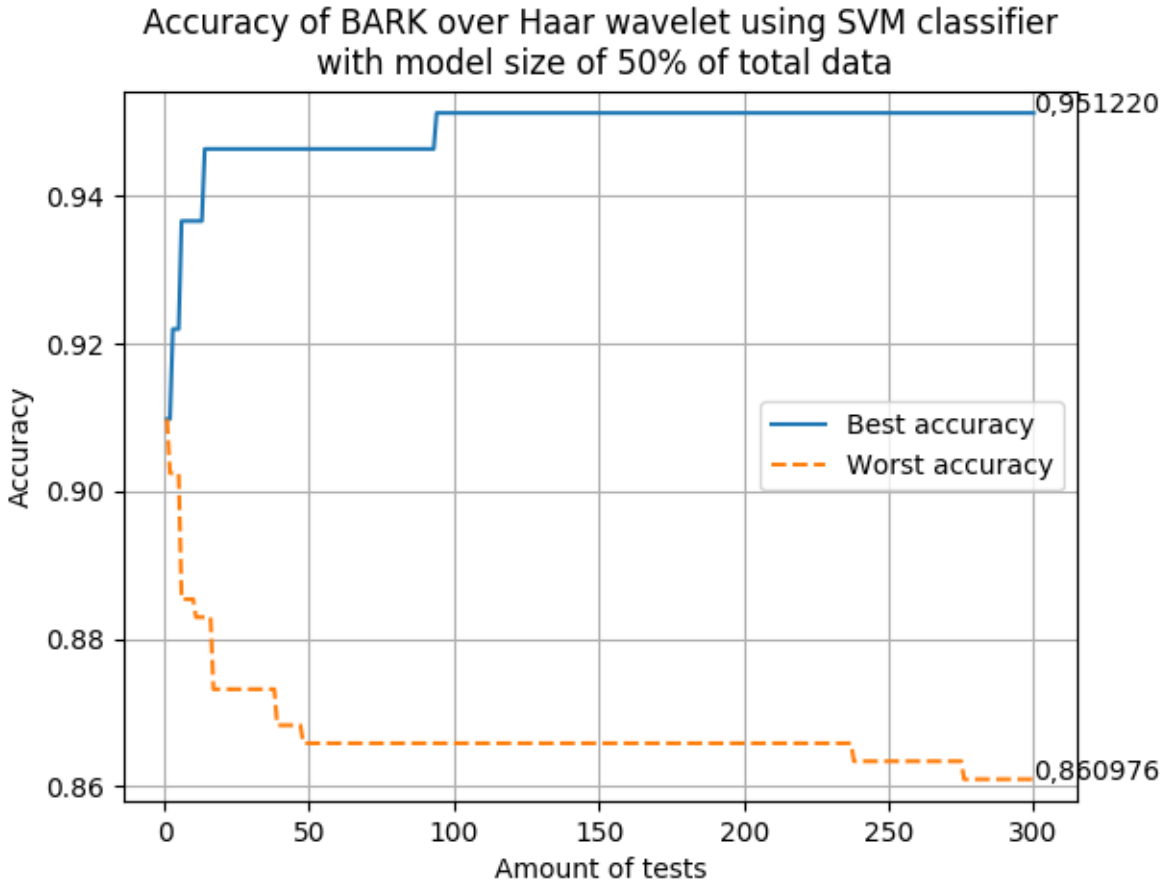


Figura 4.16: Acurácia *X* quantidade de testes - SVM, modelo a 50%

	Verdadeiro	Falso
Verdadeiro	197	12
Falso	8	193

Tabela 4.18: Tabela de confusão para classificador SVM 50%

Capítulo 5

Conclusões

5.1 subsec1

5.2 subsec2

Bibliografia

[AKY+18] A. Awais, S. Kun, Y. Yu, S. Hayat, A. Ahmed, and T. Tu. Speaker recognition using mel frequency cepstral coefficient and locality sensitive hashing. In *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 271–276, May 2018.

[AV19] KNRK Raju Alluri and Anil Kumar Vuppala. Replay spoofing countermeasures using high spectro-temporal resolution features. *International Journal of Speech Technology*, 22(1):271–281, 2019.

[AWG09] P. S. Addison, J. Walker, and R. C. Guido. Time–frequency analysis of biosignals. *IEEE Engineering in Medicine and Biology Magazine*, 28(5):14–29, Sep. 2009.

[Con18] Linguistic Data Consortium. Timit acoustic-phonetic continuous speech corpus, 2018.

[DGK+16] K. Arun Das, Kuruvachan K. George, C. Santhosh Kumar, S. Veni, and Ashish Panda. Modified Gammatone Frequency Cepstral Coefficients to Improve Spoofing Detection. pages 50–55, 345 E 47TH ST, NEW YORK, NY 10017 USA, 2016. LNM Inst Informat Technol; IEEE Commun Soc; IEEE Syst Man & Cybernet Soc, IEEE. International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, INDIA, SEP 21-24, 2016.

[Fre13] Susana Freitas. Avaliação acústica e áudio percetiva na caracterização da voz humana. 2013.

[Gui19] R. C. Guido. Paraconsistent feature engineering [lecture notes]. *IEEE Signal Processing Magazine*, 36(1):154–158, Jan 2019.

[Han18] Cemal Hanilçi. Linear prediction residual features for automatic speaker verification anti-spoofing. *Multimedia Tools and Applications*, 77(13):16099–16111, Jul 2018.

[JlCH01] Arne Jensen and Anders la Cour-Harbo. *Ripples in Mathematics*. Springer Berlin Heidelberg, 2001.

[KG14] Robinson Luis Kremer and ML d C GOMES. A eficiência do disfarce em vozes femininas: uma análise da frequência fundamental. *ReVEL*, 12:23, 2014.

[KMM+16] Pavel Korshunov, Sébastien Marcel, Hannah Muckenhirn, André R Gonçalves, AG Souza Mello, RP Velloso Violato, Flávio O Simoes, M Uliani Neto, Marcus de Assis Angeloni, José Augusto Stuchi, et al. Overview of btas 2016 speaker anti-spoofing competition. pages 1–6. IEEE, 2016.

[KP17] Madhu R. Kamble and Hemant A. Patil. Novel Energy Separation Based Frequency Modulation Features For Spoofed Speech Classification. pages 326–331, 2017. 9th International Conference on Advances in Pattern Recognition (ICAPR), Indian Stat Inst Bangalore, Bangalore, INDIA, DEC 27-30, 2017.

[KP18] Madhu R. Kamble and Hemant A. Patil. Novel Variable Length Energy Separation Algorithm using Instantaneous Amplitude Features For Replay Detection. Interspeech, pages 646–650. Int Speech Commun Assoc, 2018. 19th Annual Conference of the International-Speech-Communication-Association (INTERSPEECH 2018), Hyderabad, INDIA, AUG 02-SEP 06, 2018.

[NKL+16] S. Novoselov, A. Kozlov, G. Lavrentyeva, K. Simonchik, and V. Shchemelinin. Stc anti-spoofing systems for the asvspoof 2015 challenge. pages 5475–5479, March 2016.

[P+96] Robi Polikar et al. The wavelet tutorial, 1996.

[PP15] Tanvina Patel and Hemant Patil. Combining evidences from mel cepstral, cochlear filter cepstral and instantaneous frequency features for detection of natural vs. spoofed speech. 09 2015.

[Pro] RedDots Project. Reddots database.

[RBABA19] Raoudha Rahmeni, Anis Ben Aicha, and Yassine Ben Ayed. On the contribution of the voice texture for speech spoofing detection. pages 501–505, 345 E 47TH ST, NEW YORK, NY 10017 USA, 2019. IEEE.

[RFLC19] Yanzhen Ren, Zhong Fang, Dengkai Liu, and Changwen Chen. Replay attack detection based on distortion by loudspeaker for voice authentication. *Multimedia Tools and Applications*, 78(7):8383–8396, Apr 2019.

[Sap19] Craig Stuart Sapp. Wave pcm soundfile format, 2019.

[SPM18] M. S. Saranya, R. Padmanabhan, and Hema A. Murthy. Replay Attack Detection in Speaker Verification Using non-voiced segments and Decision Level Feature Switching. International Conference on Signal Processing and Communications SPCOM, pages 332–336. IEEE, 2018. 12th International Conference on Signal Processing and Communications (SPCOM), Indian Inst Sci, Bangalore, INDIA, JUL 16-19, 2018.

[SSAL17] K. Sriskandaraja, V. Sethu, E. Ambikairajah, and H. Li. Front-end for antispoofing countermeasures in speaker verification: Scattering spectral decomposition. *IEEE Journal of Selected Topics in Signal Processing*, 11(4):632–643, June 2017.

[SSWA18] Gajan Suthokumar, Vidhyasaharan Sethu, Chamith Wijenayake, and Eliathamby Ambikairajah. Modulation dynamic features for the detection of replay attacks. pages 691–695, 2018.

[TDE17] Massimiliano Todisco, Héctor Delgado, and Nicholas Evans. Constant q cepstral coefficients: A spoofing countermeasure for automatic speaker verification. *Computer Speech & Language*, 45:516 – 535, 2017.

[TK17] Junichi Yamagishi et al Tomi Kinnuneni, Nicholas Evans. Asvspoof 2017, 2017.

[V+14] Eugênia Hermínia Oliveira Valença et al. Análise acústica dos formantes em indivíduos com deficiência isolada do hormônio do crescimento. 2014.

[WIAE18] Buddhi Wickramasinghe, Saad Irtza, Eliathamby Ambikairajah, and Julien Epps. Frequency Domain Linear Prediction Features for Replay Spoofing Attack Detection. Interspeech, pages 661–665, C/O EMMANUELLE FOXONET, 4 RUE DES FAUVETTES, LIEU DIT LOUS TOURILS, BAIXAS, F-66390, FRANCE, 2018. Int Speech Commun Assoc, ISCA-INT SPEECH COMMUNICATION ASSOC. 19th Annual Conference of the International-Speech-Communication-Association (INTERSPEECH 2018), Hyderabad, INDIA, AUG 02-SEP 06, 2018.

[YJ19] et al Yamagishi Junichi, Todisco Massimiliano. Spoofing and anti-spoofing (sas) corpus, 2019.

[YLYL19] Y. Ye, L. Lao, D. Yan, and L. Lin. Detection of replay attack based on normalized constant q cepstral feature. pages 407–411, April 2019.

[YXWW19] Diqun Yan, Li Xiang, Zhifeng Wang, and Rangding Wang. Detection of hmm synthesized speech by wavelet logarithmic spectrum. *Automatic Control and Computer Sciences*, 53(1):72–79, Jan 2019.

[ZW15] Nicholas Evans Junichi Yamagishi Zhizheng Wu, Tomi Kinnunen. Asvspoof 2015, 2015.

Capítulo 6

Apêndices

Wavelet	G1	G2	Distância do ponto (1,0)
haar	0.93615	4.68316e-310	0.0638503
daub4	0.928088	4.68316e-310	0.0719123
daub6	0.927885	4.68316e-310	0.072115
coif6	0.927823	4.68316e-310	0.072177
sym8	0.92769	4.68316e-310	0.0723096
daub12	0.926541	4.68316e-310	0.073459
coif12	0.926479	4.68316e-310	0.0735206
daub10	0.925778	4.68316e-310	0.074222
coif18	0.925636	4.68316e-310	0.0743642
daub38	0.924105	4.68316e-310	0.0758951
daub58	0.923782	4.68316e-310	0.0762175
coif24	0.923759	4.68316e-310	0.0762411
daub32	0.923231	4.68316e-310	0.0767692
daub72	0.923049	4.68316e-310	0.076951
daub18	0.922752	4.68316e-310	0.0772476
daub76	0.922538	4.68316e-310	0.0774616
daub70	0.922464	4.68316e-310	0.0775363
daub16	0.922434	4.68316e-310	0.0775664
daub26	0.921938	4.68316e-310	0.0780617
daub20	0.921789	4.68316e-310	0.0782114
sym16	0.921534	4.68316e-310	0.0784657
daub40	0.921381	4.68316e-310	0.0786195
daub8	0.921189	4.68316e-310	0.0788111

Tabela 6.1: Combinação wavelet*X*BARK no plano paraconsistente (Parte 01)

Wavelet	G1	G2	Distância do ponto (1,0)
haar	0.93615	4.68316e-310	0.0638503
daub62	0.921129	4.68316e-310	0.0788712
daub74	0.921103	4.68316e-310	0.0788972
daub22	0.921018	4.68316e-310	0.0789818
daub68	0.920958	4.68316e-310	0.0790417
daub52	0.920935	4.68316e-310	0.0790645
daub66	0.920698	4.68316e-310	0.0793024
vaidyanathan24	0.92032	4.68316e-310	0.0796802
daub46	0.920243	4.68316e-310	0.0797574
sym32	0.920126	4.68316e-310	0.0798735
daub64	0.919111	4.68316e-310	0.0808888
daub60	919	4.68316e-310	0.0809997
daub48	0.918767	4.68316e-310	0.081233
daub24	0.918569	4.68316e-310	0.0814307
coif30	0.918532	4.68316e-310	0.0814678
daub14	0.918381	4.68316e-310	0.0816185
daub54	0.918312	4.68316e-310	0.081688
daub44	0.918082	4.68316e-310	0.0819177
daub30	0.917617	4.68316e-310	0.0823828
beylkin18	0.917392	4.68316e-310	0.0826082
daub36	0.91732	4.68316e-310	0.0826799
daub56	0.917317	4.68316e-310	0.0826829
daub50	0.917195	4.68316e-310	0.0828052
daub28	0.916716	4.68316e-310	0.0832843
daub34	0.916594	4.68316e-310	0.0834055
daub42	0.916039	4.68316e-310	0.083961

Tabela 6.2: Combinação wavelet*X*BARK no plano paraconsistente (Parte 02)

Wavelet	G1	G2	Distância do ponto (1,0)
haar	0.728548	4.68316e-310	0.271452
daub4	0.685787	4.68316e-310	0.314213
sym16	0.684548	4.68316e-310	0.315452
daub18	0.679639	4.68316e-310	0.320361
coif6	0.677144	4.68316e-310	0.322856
daub32	0.67416	4.68316e-310	0.32584
daub8	0.672315	4.68316e-310	0.327685
sym8	0.668039	4.68316e-310	0.331961
daub68	0.663504	4.68316e-310	0.336496
daub28	0.661634	4.68316e-310	0.338366
daub20	0.658861	4.68316e-310	0.341139
daub76	0.655598	4.68316e-310	0.344402
daub48	0.655121	4.68316e-310	0.344879
daub38	0.65436	4.68316e-310	0.34564
daub24	0.652545	4.68316e-310	0.347455
daub12	0.64926	4.68316e-310	0.35074
coif30	0.648829	4.68316e-310	0.351171
daub52	0.644251	4.68316e-310	0.355749
coif12	0.643978	4.68316e-310	0.356022
daub22	0.640718	4.68316e-310	0.359282
daub42	0.639246	4.68316e-310	0.360754
sym32	0.638775	4.68316e-310	0.361225
beylkin18	0.638489	4.68316e-310	0.361511

Tabela 6.3: Combinação wavelet*X*MEL no plano paraconsistente (Parte 01)

Wavelet	G1	G2	Distância do ponto (1,0)
haar	0.728548	4.68316e-310	0.271452
daub4	0.685787	4.68316e-310	0.314213
sym16	0.684548	4.68316e-310	0.315452
daub18	0.679639	4.68316e-310	0.320361
coif6	0.677144	4.68316e-310	0.322856
daub32	0.67416	4.68316e-310	0.32584
daub8	0.672315	4.68316e-310	0.327685
sym8	0.668039	4.68316e-310	0.331961
daub68	0.663504	4.68316e-310	0.336496
daub28	0.661634	4.68316e-310	0.338366
daub20	0.658861	4.68316e-310	0.341139
daub76	0.655598	4.68316e-310	0.344402
daub48	0.655121	4.68316e-310	0.344879
daub38	0.65436	4.68316e-310	0.34564
daub24	0.652545	4.68316e-310	0.347455
daub12	0.64926	4.68316e-310	0.35074
coif30	0.648829	4.68316e-310	0.351171
daub52	0.644251	4.68316e-310	0.355749
coif12	0.643978	4.68316e-310	0.356022
daub22	0.640718	4.68316e-310	0.359282
daub42	0.639246	4.68316e-310	0.360754
sym32	0.638775	4.68316e-310	0.361225
beylkin18	0.638489	4.68316e-310	0.361511

Tabela 6.4: Combinação wavelet*X*MEL no plano paraconsistente (Parte 02)

Packet wavelet for haar filter = {0.5, 0.5}

Signal	32	10	20	38	37	28	38	34	18	24	24	9	23	24	28	34
	h								g							
Level 01	21	29	32,5	36	21	16,5	23,5	31	11	-9	4,5	2	-3	7,5	-0,5	-3
	h				g				g				h			
Level 02	25	34,25	18,75	27,25	-4	-1,75	2,25	-3,75	10	1,25	-5,25	1,25	1	3,25	2,25	-1,75
	h		g		g		h		h		g		g		h	
Level 03	29,62	23	-4,625	-4,25	-1,125	3	-2,875	-0,75	5,625	-2	4,375	-3,25	-1,125	2	2,125	0,25
	h	g	g	h	h	g	g	h	h	g	g	h	h	g	g	h
Level 04	26,3125	3,3125	-0,1875	-4,4375	0,9375	-2,0625	-1,0625	-1,8125	1,8125	3,8125	3,8125	0,5625	0,4375	-1,5625	0,9375	1,1875

← **result of lowpass (h)**
← **result of highpass (g)**

Results

Regular wavelet

Signal	32	10	20	38	37	28	38	34	18	24	24	9	23	24	28	34
Level 01	21	29	32,5	36	21	16,5	23,5	31	11	-9	4,5	2	-3	7,5	-0,5	-3
Level 02	25	34,25	18,75	27,25	-4	-1,75	2,25	-3,75	11	-9	4,5	2	-3	7,5	-0,5	-3
Level 03	29,62	23	-4,625	-4,25	-4	-1,75	2,25	-3,75	11	-9	4,5	2	-3	7,5	-0,5	-3
Level 04	26,3125	3,3125	-4,625	-4,25	-4	-1,75	2,25	-3,75	11	-9	4,5	2	-3	7,5	-0,5	-3

Wavelet packet

Signal	32	10	20	38	37	28	38	34	18	24	24	9	23	24	28	34
Level 01	21	29	32,5	36	21	16,5	23,5	31	11	-9	4,5	2	-3	7,5	-0,5	-3
Level 02	25	34,25	18,75	27,25	-4	-1,75	2,25	-3,75	10	1,25	-5,25	1,25	1	3,25	2,25	-1,75
Level 03	29,62	23	-4,625	-4,25	-1,125	3	-2,875	-0,75	5,625	-2	4,375	-3,25	-1,125	2	2,125	0,25
Level 04	26,3125	3,3125	-0,1875	-4,4375	0,9375	-2,0625	-1,0625	-1,8125	1,8125	3,8125	3,8125	0,5625	0,4375	-1,5625	0,9375	1,1875

Figura 6.1: Demostração numérica das transformadas Wavelet e Wavelet packet

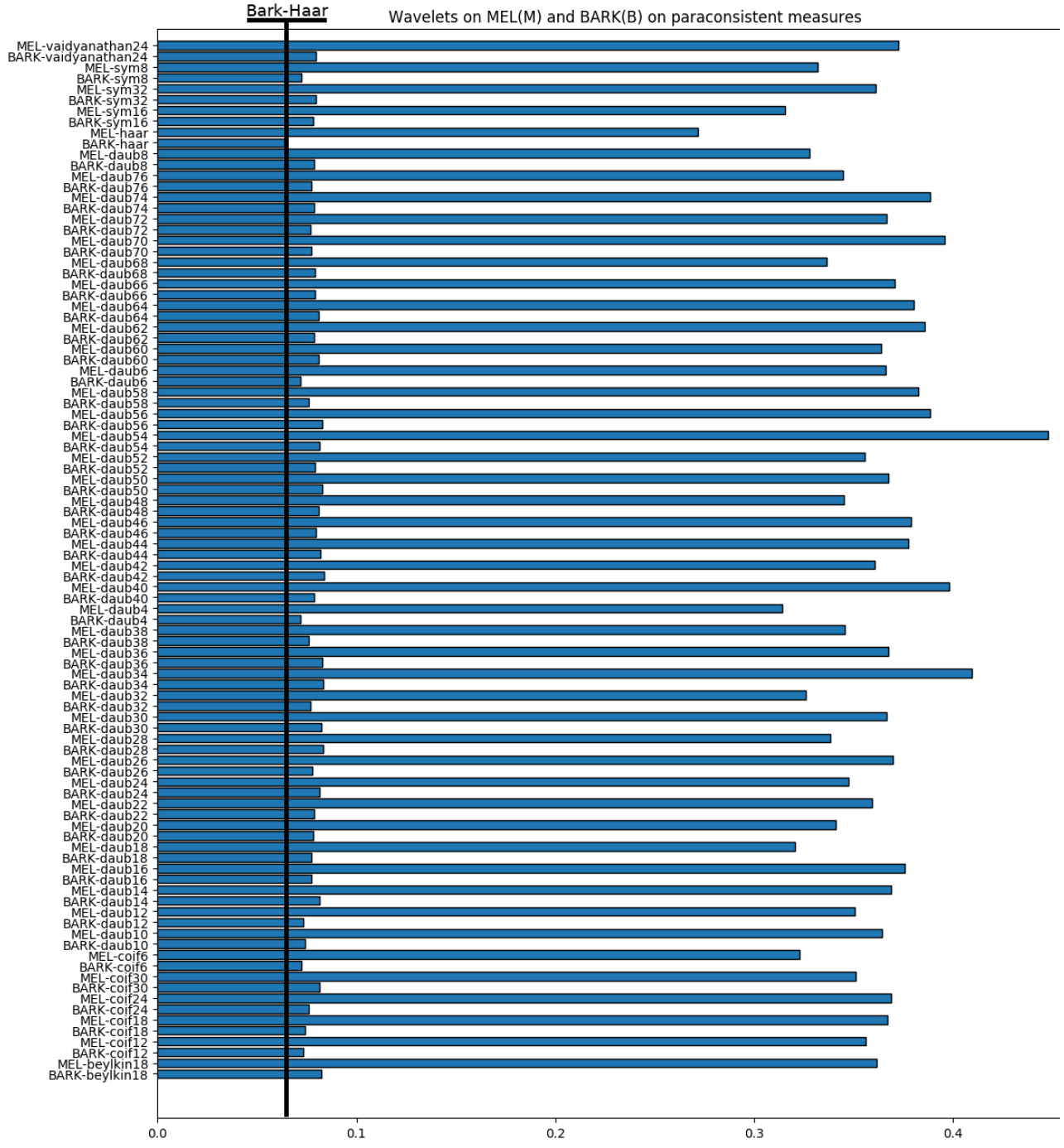


Figura 6.2: Gráfico completo da distância ao ponto (1,0) no plano paraconsistente.