



**UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

BRUNO GARCIA DA COSTA

**CLASSIFICAÇÃO DE FALTAS DO TIPO CURTO CIRCUITO EM
LINHAS DE TRANSMISSÃO UTILIZANDO KNN-DTW**

UFPA / ICEN / PPGCC

Belém - Pará

2017



UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

BRUNO GARCIA DA COSTA

CLASSIFICAÇÃO DE FALTAS DO TIPO CURTO CIRCUITO EM
LINHAS DE TRANSMISSÃO UTILIZANDO KNN-DTW

Dissertação de Mestrado apresentado ao Programa de Pós-Graduação em Ciência da Computação. Instituto de Ciências Exatas e Naturais. Universidade Federal do Pará.

Área de Concentração: Sistemas de Computação
Orientador: Prof. Dr. Jefferson Magalhães de Moraes.

UFPA / ICEN / PPGCC

Belém - Pará

2017

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

C837c Costa, Bruno Garcia
CLASSIFICAÇÃO DE FALTAS DO TIPO CURTO CIRCUITO EM LINHAS DE TRANSMISSÃO
UTILIZANDO KNN-DTW / Bruno Garcia Costa. - 2017.
63 f. : il. color.

Dissertação (Mestrado) - Programa de Pós-graduação em Ciência da Computação (PPGCC), Instituto de Ciências Exatas e Naturais, Universidade Federal do Pará, Belém, 2017.
Orientação: Prof. Dr. Jefferson Magalhães Morais

1. Sistema Elétrico de Potência. 2. Classificação de Faltas. 3. Arquitetura FBSC. 4. Algoritmo KNN-DTW. I. Morais, Jefferson Magalhães, *orient.* II. Título

CDD 004.24

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

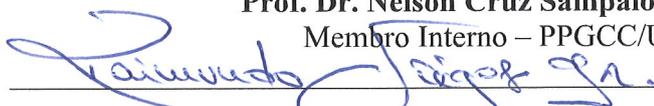
BRUNO GARCIA DA COSTA

**CLASSIFICAÇÃO DE FALTAS DO TIPO CURTO CIRCUITO EM LINHAS DE
TRANSMISSÃO UTILIZANDO KNN-DTW**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Pará como requisito para obtenção do título de Mestre em Ciência da Computação, defendida e aprovada em 15/12/2017, pela banca examinadora constituída pelos seguintes membros:


Prof. Dr. Jefferson Magalhães de Moraes
Orientador – PPGCC/UFPA


Prof. Dr. Nelson Cruz Sampaio Neto
Membro Interno – PPGCC/UFPA


Prof. Dr. Raimundo Viegas Junior
Membro Externo – FACOMP/UFPA

Visto: 
Prof. Dr. Bianchi Serique Meiguins
Coordenador do PPGCC/UFPA

À minha namorada...
E aos meus pais...
Com todo amor e carinho.

AGRADECIMENTOS

Agradecer é o que mais me faz feliz nessa causa tão nobre de alcançar o processo final do Mestrado, então em primeiro lugar vem minha gratidão a Deus, por me conceder esse momento inesquecível com saúde e felicidade pela conquista alcançada.

Ao meu orientador, Prof. Dr Jefferson Moraes, apresento meu sincero agradecimento, pois sempre foi constante em suas decisões, desde a primeira conversa que tivemos, sempre contribuiu com discussões técnicas e precisas para conclusão desse trabalho, sempre acreditando no meu desenvolvimento acadêmico e profissional.

Ao Programa de Pós-Graduação em Ciência da Computação (PPGCC/UFGA), em especial ao corpo docente do Programa, que me ofereceu a oportunidade de aprendizado de alto nível, que usarei por toda minha vida.

Aos meus pais, Terezinha Garcia e Manoel Pereira, que demonstraram o mesmo carinho em todas as fases da minha vida, acreditando, motivando e dando todo suporte dentro de suas possibilidades.

Aos amigos e companheiros que foram conquistados durante esse processo, em especial ao Prof. Dr Raimundo Viégas, que compartilhou um pouco da sabedoria e experiência adquirida no meio acadêmico, sem esquecer os que contribuíram com a pesquisa como Hamilton Cavalcante e outros discentes que cursaram algumas disciplinas juntos.

Ao Governo do Amapá, pelo o incentivo concedido nos anos que foram cursado o Mestrado, sou grato pelo apoio das secretarias do estado do Amapá, em especial ao Centro de Gestão da Tecnologia da informação (PRODAP).

A todos aqueles que não foram citados, mas que tiveram participação na conclusão desta etapa da minha vida.

*“Talvez não tenha conseguido fazer o melhor, mas lutei para que o melhor fosse feito.
Não sou o que deveria ser, mas Graças a Deus, não sou o que era antes”.*
(Marthin Luther King)

SUMÁRIO

1	INTRODUÇÃO	15
1.1	Motivação e descrição geral do problema	15
1.2	Objetivos	17
1.2.1	Objetivo Geral	17
1.2.2	Objetivos Específicos	18
1.3	Trabalhos relacionados	18
1.4	Contribuições do trabalho	21
1.5	Trabalhos publicados	22
1.6	Estrutura do trabalho	22
2	FUNDAMENTAÇÃO TEÓRICA	23
2.1	Introdução	23
2.2	Classificação de sequências representando curto-circuitos em linhas de transmissão	24
2.3	Classificação de sequências baseados em quadros	25
2.4	Weka	26
2.5	Classificadores convencionais	26
2.5.1	Redes neurais artificiais	27
2.5.2	Árvores de decisão	28
2.5.3	Máquinas de vetores de suporte	30
2.5.4	K-vizinhos mais próximos	30
2.6	Classificação de sequências baseada no classificador KNN-DTW	31
3	METODOLOGIA PROPOSTA	37
3.1	Base de dados	37
3.2	Front ends	39
3.2.1	Direto da forma de onda - <i>Raw</i>	39
3.2.2	<i>Front end</i> RMS	40
3.2.3	<i>Front ends wavelets</i>	41
3.2.4	Concatenação de <i>fronts ends</i>	45
4	AVALIAÇÃO EXPERIMENTAL DA PROPOSTA	47
4.1	Configuração dos experimentos	47
4.2	Resultados para classificação de sequências utilizando KNN-DTW	48
4.3	Comparando a Arquitetura FBSC com o classificador KNN-DTW	50
5	CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS	53
5.1	Trabalhos Futuros	54
	REFERÊNCIAS	55
	ANEXO A – CÓDIGO DO KNN-DTW EM JAVA	60

LISTA DE ILUSTRAÇÕES

Figura 1 – Representação do Sistema Elétrico de Potência (SEP)	23
Figura 2 – Processo de classificação direta de sequências	24
Figura 3 – Fluxo de processamento da arquitetura FBSC. A saída do classificador de sequência $G(Z)$ depende das decisões do classificador $F(z_n)$	25
Figura 4 – Estrutura de um neurônio artificial	27
Figura 5 – Exemplo de uma árvore de decisão.	29
Figura 6 – Monotocidade	32
Figura 7 – Continuidade	33
Figura 8 – Condição de contorno	33
Figura 9 – Alinhamento de janela	34
Figura 10 – Restrição de inclinação	34
Figura 11 – Sequência encontrada após o cálculo DTW	35
Figura 12 – Arquivo ASCII representando as informações associadas disponibilizada na base UFPAFaults.	38
Figura 13 – Demonstração do <i>front em raw</i> , organizando os vetores de padrão \mathbf{z} . Neste exemplo, $L = 2$ e $e = 2$ (não há sobreposição) e a matriz obtida pela aplicação do <i>front end</i> possui dimensão $K = 12$ com quadros da falta “ABT” e um quadro falta “BC”.	40
Figura 14 – Filtragem de um estágio para geração de aproximações (A) e detalhes (D) de um sinal (S).	42
Figura 15 – Exemplo de decomposição de 3 níveis.	42
Figura 16 – Decomposição wavelet a partir da função wavedec do MATLAB.	43
Figura 17 – Exemplo de uma decomposição <i>wavelet</i> aplicada ao sinais de tensão das fases A e B de uma falta AB simulada no intervalo de 1s. A <i>wavelet</i> mãe é uma <i>Daubechies</i> 4 com três níveis de decomposição ($\gamma = 3$).	44
Figura 18 – Processo de concatenação dos <i>front ends</i> , mostrando o número de parâmetros K para cada <i>front end</i> , considerando $L_{min} = 9$ e $S_{min} = 4$	46
Figura 19 – Curva de complexidade amostral para o classificador KNN-DTW, considerando 1 vizinho mais próximo.	48
Figura 20 – Custo computacional utilizando DTW	49
Figura 21 – Avaliação da robustez do classificador KNN-DTW em termos de taxa de erro sob uma condição de razão sinal ruído (SNR - <i>Signal-to-Noise Ratio</i>) variando em 10, 20, 30, 40 e 50 dB. O conjunto de treino foi composto por 1000 amostras sem ruído.	50
Figura 22 – Comparação das taxas de erro dos classificadores convencionais da arquitetura FBSC com o classificador KNN-DTW. Os valores de $L = L_{min}$ e $S = S_{min}$ dos <i>fronts ends</i> testados foram 9 e 4, respectivamente	51

Figura 23 – Comparação, em termos de acurácia (taxa de erro), da arquitetura FBSC e do classificador KNN-DTW em um cenário com ruído. O classificador da arquitetura FBSC utilizado foi o KNN convencional que utiliza a distância euclidiana como medida de similaridade. Os **front ends** usados foram waveletconcat, waveletenergy, Raw, RMS e concatfrontend, considerando $L = L_{min} = 9$ e $S = S_{min} = 4$ 52

LISTA DE TABELAS

Tabela 1 – Exemplos de distúrbios relacionados à QEE e suas respectivas causas e efeitos.	16
Tabela 2 – Ilustração do cenário dos trabalhos relacionados.	21

LISTA DE ABREVIATURAS E SIGLAS

ANN	<i>Artificial Neural Network</i> – Rede Neural Artificial
ARFF	<i>Attribute-Relation File Format</i>
ATP	<i>Alternative Transient Program</i>
DTW	<i>Dynamic Time Warping</i>
DWT	<i>Discrete Wavelet Transform</i> – Transformada Discreta de Wavelet
FBSC	<i>Frame Based-Sequence Classification</i> – Classificador de Sequencia Baseado em Quadros
HMM	<i>Hidden Markov Models</i> – Cadeias Ocultas de Markov
IBK	<i>Instance-Bases Learning With Parameter K</i>
KNN	<i>K-Nearest Neighbors</i> – K-vizinhos mais próximos
MLP	<i>Multilayer Perceptron</i>
PCA	<i>Principal Component Analysis</i> – Componente de Análise Principal
QEE	Qualidade de Energia Elétrica
RBF	<i>Radial Basis Functions</i> – Funções de Base Radial
RMS	<i>Root Mean Square</i>
SEP	Sistemas Elétricos de Potência
SVM	<i>Support Vector Machine</i> – Máquinas de Vetores de Suporte
WEKA	<i>Waikato Environment for Knowledge Analysis</i>
$x(t)$	Forma de onda de tensão, $t = 1, \dots, T$
K	Número de linhas de Z , onde $K = Q \times L$
Z	Matriz chamada “ <i>instance</i> ”, a qual corresponde à matriz \hat{Z} redimensionada, de maneira a ter dimensão $K \times N_n$, por conveniência
f_s	Frequência de amostragem
L	Número de amostras (tamanho) do quadro (<i>frames</i>)
N_n	Número total de quadros

X_n	Matriz de dimensão $Q \times T_n$ representando o n-ésimo evento em uma base de dados
T_n	Número de amostras multidimensionais do n-ésimo evento
Q	Número de sinais de tensão e corrente nas fases A, B e C
y	Rótulo ou classe, a qual corresponde à saída de um classificador
G	Classificador de sequências
F	Classificador Convencional
F	Matriz $Q \times L$ chamada quadro (<i>frame</i>), a qual aglutina L amostras de X
\hat{Z}	Matriz $Q \times L_N$ correspondente à concatenação de todos os quadros de X
S	Deslocamento do quadro: número de amostras entre as amostras de início de quadros consecutivos
E_s	Taxa de erro de classificação para o módulo de sequência (pós-falta)
R	Número de exemplos em um conjunto de teste.
I	Função indicador
K	Número de vizinhos mais próximos no classificador KNN
$z[n]$	n-ésimo valor RMS
a	Coefficiente de aproximação da transformada <i>wavelet</i>
d	Coefficiente de detalhe da transformada <i>wavelet</i>
Y	Número de estágios de filtragem e decimação de uma decomposição <i>wavelet</i> diádica.
L_{min}	Tamanho do quadro para os coeficientes <i>wavelet</i> com menor f_s
S_{min}	Deslocamento para os coeficientes <i>wavelet</i> com menor f_s
T_a	Número de amostras no coeficiente de aproximação a

RESUMO

Os sistemas de classificação de faltas em linha de transmissão podem ser divididos em dois tipos: sistemas de classificação *on-line* e pós-falta. Este trabalho foca na classificação de sequências representando faltas do tipo curto-circuito em linhas de transmissão em um cenário pós-falta. Neste cenário, as sequências possuem comprimento (duração) variável. Na classificação de sequências, é possível utilizar classificadores convencionais tais como Redes Neurais Artificiais. Neste caso, o processo de classificação requer um pré-processamento, ou um estágio de *front end*, que converta os dados brutos em parâmetros sensíveis para alimentar o classificador. Uma solução para este problema é a arquitetura de classificação de sequências baseada em quadros (FBSC). O problema da arquitetura FBSC é que esta possui muitos graus de liberdade na concepção do modelo (*front end* mais classificador) devendo este ser avaliado usando conjunto de dados completo e uma metodologia rigorosa para evitar conclusões tendenciosas. Alternativamente à arquitetura FBSC, este trabalho propõe o classificador K-*Nearest Neighbor* associado a medida de similaridade *Dynamic Time Warping* (DTW) denominado de KNN-DTW para classificação de faltas. Este classificador é capaz de lidar diretamente com as sequências de tamanhos diferentes, diminuindo o *pipeline* imposto pela arquitetura FBSC. Nos experimentos, foram utilizados dados simulados de faltas do tipo curto-circuito, oriundos de uma base de dados pública chamada UFPAFaults. Foi realizada uma comparação em termos de acurácia (taxa de erro) entre o classificador KNN-DTW e a arquitetura FBSC. No caso da arquitetura FBSC diferentes *front ends* (ex. *wavelet* e *rms*) e classificadores convencionais (ex. máquinas de vetores de suporte e redes neurais) foram testados. Os resultados obtidos indicam que em um cenário com ruído e com a necessidade de um classificador com baixa parametrização, o algoritmo KNN-DTW mostra-se como uma alternativa viável, mesmo que sua acurácia tenha sido inferior se comparado à algumas técnicas da arquitetura FBSC.

Palavras-chave: Sistema Elétrico de Potência, Classificação de Faltas, Arquitetura FBSC, Algoritmo KNN-DTW.

ABSTRACT

Fault classification system in transmission lines can be divided into two types: online and post-fault classification systems. This work focuses on the classification of sequences represented short-circuit faults in the transmission lines in a post-fault scenario. In this scenario, the sequences have variable size(duration). In sequence classification, it is possible to use conventional classifiers such as Artificial Neural Networks. In this case, the classification process requires a pre-processing or a front-end stage that converts the raw data into sensitive parameters to feed the classifier. One solution to this problem is frame-based sequence classification (FBSC). The problem of the FBSC architecture is that it has many degrees of freedom in the design of the model (front end plus classifier) and should be evaluated using a complete data set and a rigorous methodology to avoid biased conclusions. As an alternative to the FBSC architecture, this work proposes the K-Nearest Neighbor classifier associated with the Dynamic Time Warping (DTW) similarity measure, called the KNN-DTW, for fault classification. This classifier is capable of dealing directly with sequences of different sizes, reducing the pipeline imposed by the FBSC architecture. In the experiments, simulated short-circuit fault data from a public database called UFPAFaults should be used. A comparison was made in terms of accuracy between the KNN-DTW classifier and the FBSC architecture. In the case of the FBSC architecture different front ends (eg Wavelet and rms) and conventional classifiers (eg, support vector machines and neural networks) were tested. The results indicate that in a noise scenario and with the need for a classifier with low parameterization, the KNN-DTW algorithm is shown as a viable alternative, even if its accuracy has been lower. compared to some FBSC architecture techniques.

Keywords: Electrical Power System, Fault Classification, FBSC Architecture, KNN-DTW Algorithm.

1 INTRODUÇÃO

1.1 Motivação e descrição geral do problema

Sem energia elétrica a vida civilizada que conhecemos hoje seria inviável. A falta de energia elétrica paralisa as indústrias, as cidades, os transportes e as comunicações. Por isso, os sistemas elétricos de potência (SEP) têm um papel fundamental na sociedade moderna. Tais sistemas devem garantir a qualidade de energia elétrica (QEE) entregue aos consumidores. Contudo, isto nem sempre é possível, devido às faltas (distúrbios) que ocorrem na operação desses sistemas.

Os SEP são constituídos de componentes responsáveis pela geração, transmissão e distribuição de energia elétrica, os quais estão expostos a condições adversas e imprevisíveis, fazendo-se necessário a utilização de sistemas de monitoramento para analisar o comportamento do sinal elétrico com objetivo de garantir a QEE fornecida pelas concessionária de energia elétrica. Os diversos distúrbios que provocam alterações nos valores de tensão, corrente ou desvio de frequência, resultam em faltas que comprometem a QEE. Os principais distúrbios de QEE que ocorrem nos SEP são apresentados na Tabela 1.

A linha de transmissão é o componente de um sistema elétrico de potência mais vulnerável a faltas, especialmente se considerarmos suas dimensões físicas, complexidade estrutural e o ambiente em que se encontram. Dentre as faltas que podem ocorrer em uma linha de transmissão, as faltas do tipo curto-circuito são as que provocam maiores impactos para o consumidor. Estudos revelaram que as faltas são responsáveis por cerca de 70% dos distúrbios ocorridos nos sistemas elétricos (CHATTOPADHYAY; MITRA; SENGUPTA, 2011; ZHANG; KEZUNOVIC, 2007). Assim, fica evidente a necessidade dos SEP adotarem mecanismos de classificação de faltas, dando suporte à tomada de decisão a nível operacional, visando o restabelecimento do funcionamento do SEP.

Atualmente, as indústrias do setor elétrico, contam com equipamentos de tecnologia avançada que são usados na classificação dos distúrbios. Um exemplo típico dessa tecnologia, são os equipamentos de oscilografia que implementam algoritmos simples de detecção de formas de onda de tensão e corrente, capturando os valores que desviam da faixa de amplitude nominal, armazenando os eventos de interesse junto com informações adicionais como data e hora.

Apesar da tecnologia avançada, gerar resultados confiáveis e relevantes na área de classificação de faltas não é uma tarefa fácil, pois a ausência de *benchmarks* (resultados de experimentos disponíveis para comparação), influencia diretamente nas pesquisas que avaliam algoritmos de mineração de dados. Por esse motivo, as empresas do setor elétrico, não se beneficiam plenamente com o processamento automático da informação. Além disso, as bases de dados citadas na literatura são invariavelmente privadas, situação que estimulou este trabalho a utilizar a base de dados de domínio público, denominada UFPAFaults (MORAIS et al., 2010).

Tabela 1 – Exemplos de distúrbios relacionados à QEE e suas respectivas causas e efeitos.

Distúrbio	Causas	Efeitos
Afundamento da tensão	degrau de carga, curto-circuito	perda de potência, falha de operação
Flutuação da tensão	cargas variáveis, oscilação de potência	cintilação, modulação de torque
Descarga eletrostática	Sobretensões	ruptura de isolantes, sobrecorrentes
Desequilíbrio de tensões	cargas desiguais, curto entre fases	sobretensão, sobrecorrente, vibração em máquinas
Elevação da tensão	redução da carga, excesso de reativos, curto desequilibrado	estresse de dielétrico, sobrecarga
Harmônicas	cargas não lineares, chaveamento, descontinuidades	ressonância, perdas adicionais, ruído, aquecimento, interferência telefônica
Interferência eletromagnética	mau contato, chaveamento em alta frequência	aquecimento localizado, falha de sistemas digitais
Inter harmônicas	cargas não lineares variáveis	modulação harmônica, interferências
Interrupção permanente	manutenção programada, falha imprevista	parada de produção
Modulação de amplitude	carga cíclica, ressonância	cintilação, oscilação de torque
Oscilação eletromecânica	desbalanço entre geração e carga	modulação da potência gerada, variação da frequência
Ressonância subsíncrona	compensação capacitiva série de linha	vibração mecânica entre turbina e gerador
Sobretensão	curto desequilibrado, entrada de capacitor, redução de carga	ruptura de dielétricos, sobrecorrente, aumento de perdas, queima de aparelhos, redução da vida útil
Surto de tensão	descarga eletrostática, curto circuito	ruptura de dielétrico, queima de equipamentos

Fonte: (HOMCI, 2016)

Os sistemas de classificação de faltas em linha de transmissão podem ser divididos em dois tipos: sistemas de classificação *on-line* e pós-falta (ZHONGMING; BIN, 2000). Os sistemas de classificação *on-line* realizam uma decisão (classificação) em um curto espaço de tempo, com o segmento de análise (ou quadro) estando localizado aproximadamente no instante em que a falta ocorre. A classificação pós-falta pode ser executada *off-line* e sua entrada consiste em uma série temporal multivariada com comprimento (duração) variável, diferente da classificação *on-line* em que a entrada é um vetor de tamanho fixo. Os sistemas *on-line* e pós-falta tentam

resolver problemas que podem ser tratados como problemas de classificação convencional e de sequência (SUDHA, 2007), respectivamente. Este trabalho foca na classificação de sequência em um cenário pós-falta.

Na classificação de sequência (pós-falta), é possível utilizar classificadores convencionais tais como Redes Neurais Artificiais e Máquinas de Vetores de Suporte (*Support Vector Machine* - SVM) (SINGH; PANIGRAHI; MAHESHWARI, 2011). Neste caso, o processo de classificação requer um pré-processamento ou um estágio de *front end* que converta os dados brutos em parâmetros sensíveis para alimentar o *back-end* (neste caso, o classificador).

Em alguns trabalhos na literatura, transformada *wavelet* tem sido utilizada com outras técnicas tais como ANN, SVM e Sistemas *Fuzzy*, compondo *frameworks* híbridos para classificação de faltas (SAWATPIPAT; TAYJASANANT, 2010). Utilizam tais *front ends*, como Transformada de *Fourier*, *root mean square* (RMS) e *raw*, bastante utilizados na classificação de faltas em linhas de transmissão (ABDOLLAHI; SEYEDTABAIL, 2010).

Em (MORAIS et al., 2010), foi proposta a arquitetura *frame based sequence classification* (FBSC) para classificação de faltas em linhas de transmissão. O principal método da arquitetura FBSC é segmentar a sequência de entrada em vetores de tamanho fixo, denominado de quadros (*frames*) e invocar repetidamente um classificador convencional para processar cada quadro. O classificador FBSC considera as saídas do classificador convencional e a partir de um processo de *voting* chega a uma decisão final, observando qual a classe mais frequente. O problema da arquitetura FBSC é que esta possui muitos graus de liberdade na concepção do modelo (*front end* mais classificador), devendo este ser avaliado usando um conjunto de dados completo e uma metodologia rigorosa para evitar conclusões tendenciosas.

Alternativamente à arquitetura FBSC, este trabalho propõe a adoção de um classificador capaz de lidar diretamente com as sequências, diminuindo o *pipeline* (custo computacional para classificar sequências, vide figura 3) imposto pela arquitetura FBSC. Mais especificamente, será utilizado o classificador *K-Nearest Neighbor* (KNN) (FACELI, 2011) associado a medida de similaridade *Dynamic Time Warping* (DTW), que permite comparar sequências de tamanhos diferentes. Este classificador é denominado na literatura de KNN-DTW, sendo bastante utilizado em classificação de series temporais (PETITJEAN et al., 2014).

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral deste trabalho, consiste em classificar faltas do tipo curto-circuito (pós-falta) em linhas de transmissão utilizando o algoritmo KNN-DTW. Para isso, será utilizada a base de dados UFPAFaults e os resultados da acurácia deste classificador serão comparados com os da arquitetura FBSC.

1.2.2 Objetivos Específicos

- Apresentar as características mais relevantes da base de dados UFPAFaults para obter resultados satisfatórios e confiáveis;
- Testar o algoritmo KNN-DTW para classificação de faltas em linha de transmissão observando sua acurácia utilizando a base de dados UFPAFaults. Tanto cenários com ruídos quanto sem ruídos serão analisados;
- Replicar os resultados da arquitetura FBSC para classificação de faltas conforme a metodologia apresentada em (MORAIS et al., 2010);
- Comparar os resultados obtidos em termos de acurácia pelo classificador KNN-DTW com os da arquitetura FBSC.

1.3 Trabalhos relacionados

Trabalhos recentes da literatura mostram a utilização de técnicas de pré-processamento ou *front ends* (ex. transformada *wavelet* e RMS), para solucionar o problema de classificação de séries temporais multivariadas, representando faltas do tipo curto-circuito em linhas de transmissão, visto que, as faltas em relação ao tempo apresentam tamanhos diferentes, impossibilitando o reconhecimento de padrões por classificadores convencionais (ex. ANN). A seguir, são apresentados alguns trabalhos relacionados que utilizam diferentes tipos de *front ends* e classificadores na tarefa de classificação de faltas.

Em (CARDOSO et al., 2008), é proposta uma metodologia baseada no algoritmo de agrupamento hierárquico aglomerativo (HAC) para rotular as faltas (curto-circuitos) que ocorrem em linhas de transmissão. Foram utilizados dois *front ends*: Análise de Componentes Principais (*Principal Componente Analysis* - PCA) e *Root Mean Squared* (RMS). Além disso, duas medidas de distâncias foram adotadas: distância euclidiana para o *front end* RMS e distância de Eros para o *front end* PCA. Nas simulações foi utilizada a base de dados UFPAFaults e cinco métodos aglomerativos foram testados: *Single link* (SL), *Complete link* (CL), *Weighted pair group average* (WPGMA), *Un-weighted pair-group method using arithmetic averages* (UPGMA) e *WARD*. Os resultados indicaram que, a menor taxa de erro (3%) obtida foi usando o método *WARD* e o *front end* PCA.

Em (ABDOLLAHI; SEYEDTABAI, 2010), é apresentada uma análise comparativa da eficiência das abordagens transformada discreta de Fourier (*Discrete Fourier Transform* - DFT) e transformada *wavelet* discreta (*Discrete Wavelet Transform* - DWT), aplicadas a detecção e classificação de faltas em linhas de transmissão. O classificador utilizado foi uma rede neural *multilayer perceptron* treinada com algoritmo *backpropagation*. Nos experimentos foi simulado um sistema de potência de 230 kV usando o simulador PSCAD/EMTDC e vários tipos de faltas, considerando diferentes condições e parâmetros. Os resultados obtidos mostraram que a

abordagem DWT apresenta melhores resultados do que DFT para detecção e classificação de faltas do tipo curto-circuito em linhas de transmissão considerando diferentes cenários de teste.

Em (SAMANTARAY, 2013), é proposta uma metodologia que combina regras baseada em lógica *Fuzzy* e o *front end S-transform*. Mais especificamente, após aplicar o *front end S-transform* para extrair as características (*features*) dos sinais de corrente da falta detectada, um classificador inicializado por uma árvore decisão e composto por regras *fuzzy* é utilizado para classificar 10 tipos de faltas. O modelo do sistema de potência usado nos experimentos foi simulado utilizando o Matlab. O modelo consiste de duas áreas de geração de 400 kV conectados por uma linha de transmissão de 300 km. Foram simulados diferentes parâmetros da linha de transmissão e o modelo adotado foi simulado com uma frequência de amostragem de 1 kHz (20 amostras por ciclo) e frequência fundamental de 50 Hz. Nos resultados, é feita uma comparação no uso do *front end wavelet* e *S-transform*, sendo que o classificador proposto denominado de DT-Fuzzy (*Decision Tree - fuzzy*) considerando a *S-transform* obteve uma taxa de erro de aproximadamente de 2%.

Em HOSSEINI(2015), assim como em (SINGH; PANIGRAHI; MAHESHWARI, 2011), é proposto um modelo híbrido para classificação e localização de faltas do tipo curto-circuito em linhas de transmissão. Na classificação é utilizado transformada *wavelet* discreta (DWT) para extrair os parâmetros iniciais das amostras das formas de onda de tensão das três fases. A *wavelet* mãe utilizada foi a *Daubechies 4* (db-4), considerando apenas o primeiro nível de decomposição *wavelet*. Como classificador foi utilizado SVM com *kernel* função de base radial Gaussiana. A taxa de erro de classificação do método proposto WT-SVM foi de 0,48%, diferente do trabalho de (SINGH; PANIGRAHI; MAHESHWARI, 2011), que obteve variações chegando a 10% de taxa de erro dependendo do cenário escolhido.

Em (GOWRISHANKAR M.; NAGAVENI, 2016), foi proposto um modelo utilizando um *front end wavelet* combinado com Rede neural (DWT-ANN) para detecção e classificação de falhas. Várias faltas na linha de transmissão são simuladas no MATLAB 8.2. O sinal de corrente trifásica da linha de transmissão é decomposto até o quinto nível de detalhe usando a transformada *wavelet* para obter a extração de características. O recurso extraído pelo processamento da transformada de *wavelet* discreta é o valor máximo e mínimo de coeficiente de detalhe de d4 e d5, que são usados para a detecção e classificação de falhas. A rede neural *multilayer perceptron* é apresentada para classificação de faltas, tendo 9,6% de taxa de erro.

Em (FATHABADI, 2016), é proposto um *framework* híbrido composto por um filtro de resposta de impulso finito (*Finite Impulse Response - FIR*) de dois estágios, quatro *support vector machines* (SVMs) e onze *support vector regressions* (SVRs) para classificação e localização de faltas em linhas de transmissão. O *framework* proposto precisa de poucas amostras para treinar as SVMs e as SVRs. Nos experimentos foi utilizado um sistema de potência trifásico de 230 kV simulado no *software* Proteus 6. O sistema de potência simulado possui uma frequência de 50 Hz e inclui uma linha de transmissão trifásica de 50 km de comprimento. Nas simulações do sistema parâmetros da linha de transmissão tais como, impedância e resistência foram variados.

As três primeiras SVMs foram treinadas para detectar os curtos-circuitos envolvendo as fases R, S e T¹, enquanto que a quarta SVM foi treinada para detectar se o curto-circuito envolveu o terra. As SVRs foram utilizadas para localização das faltas. A taxa de erro do modelo proposto foi de aproximadamente 0%.

Em (HOMCI et al., 2016), é apresentado um novo *front end*, denominado *concatfrontend*, que integra as amostras dos *front ends* em um único conjunto de sequências. Contudo, este processo de concatenação gera um conjunto de dados de alta dimensionalidade que pode provocar um aumento no custo computacional dos classificadores. Neste caso, outra contribuição do trabalho é aplicar técnicas de seleção de parâmetros para reduzir a dimensionalidade dos dados que serão utilizados no estágio subsequente da classificação de faltas. Nos experimentos, foi adotada a base de dados UFPAFaults e a arquitetura FBSC proposta em (MORAIS et al., 2010) considerando os *front ends* raw, RMS, *wavelet* e a concatenação destes. Os classificadores utilizados foram SVM, *random forest*, rede neural e K-vizinho mais próximo (KNN). Os experimentos realizados mostraram que, na maioria dos casos, o *concatfrontend* proporcionou menores taxas de erro que os outros *front ends* utilizados separadamente. Na proposta do trabalho em adotar *feature selection* para reduzir a dimensionalidade dos dados de entrada após a concatenação dos *front ends* e com isso reduzir o custo computacional dos classificadores, aplicou-se técnicas simples do tipo filtro e foi possível obter taxas de erros inferiores com um menor número de parâmetros. Vale destacar o desempenho dos classificadores rede neural e *Random Forest* nos testes feitos, se mostrando regulares e mantendo uma taxa de erro bem baixa em relação aos outros classificadores.

O trabalho de (MORAIS et al., 2010), utilizado como principal referência desta dissertação e brevemente comentado anteriormente, apresenta uma arquitetura de classificação de faltas baseada na extração de curtos segmentos (quadros - *frames*) das formas de onda de tensão e corrente armazenadas na base de dados UFPAFaults. Os resultados deste trabalho serão replicados nesta dissertação para efeitos de comparação com classificador KNN-DTW proposto.

A Tabela 2 expõe uma visão geral dos *fronts ends* e classificadores adotados pelos trabalhos relacionados a esta dissertação.

¹ Nesta dissertação as fases são chamadas de A, B e C

Tabela 2 – Ilustração do cenário dos trabalhos relacionados.

REFERÊNCIAS	TÉCNICAS UTILIZADAS												
	Front ends						Classificadores Convencionais					Classificadores Não Convencionais	
	Raw	RMS	PCA	Wavelet	Concat. Param.	S-transform	ANN	SVM	J4.8	KNN	Fuzzy	KNN DTW	FBSC
HOMCI et al, 2016	X	X		X	X		X	X		X			X
FATHABADI, 2016								X					
CARDOSO et al, 2008		X	X										
SINGH, PANIGRAHI E MAHESHWARI, 2011		X		X				X					
SAMANTARAY, 2013						X			X		X		
HOSSEINI, 2015				X				X					
ABDOLLAHI; SEYEDTABAH, 2015				X			X						
MORAIS et al, 2010	X	X		X			X	X	X	X			X
GOWRISHANKAR, NAGAVENI, E BALAKRISHNAN, 2016				X			X						

Fonte: Elaborado pelo autor.

1.4 Contribuições do trabalho

A seção anterior, referente aos trabalhos relacionados (vide Tabela 2), mostrou que a maioria dos trabalhos utilizam diferentes tipos *front ends* para lidar com o problema de sequências de tamanho variado. A aplicação dos *front ends* nas formas de onda de tensão e corrente, servem como uma etapa de pré-processamento com o objetivo de organizar os dados de entrada para posteriormente serem processados pelos classificadores convencionais que lidam somente com vetores de tamanho fixo. O problema, é a existencia de vários *front ends* que podem ser adotados, porém apresentam um grau de liberdade muito grande na escolha dos seus parâmetros, tornando o processo de escolha do *front end* uma tarefa custosa e não trivial.

Nesta dissertação, a principal contribuição consiste em utilizar o classificador KNN-DTW, capaz de lidar diretamente com as formas de onda de tensão e corrente (sequências)

de tamanho variado, sem a necessidade do uso de um *front end* para classificar faltas do tipo curto-circuito em linhas de transmissão. A aplicação desse classificador reduz o *pipeline* imposto pela arquitetura de classificação de sequências baseada em quadros (FBSC), sendo que uma comparação das suas acurácias são apresentadas e discutidas ao longo do trabalho.

1.5 Trabalhos publicados

COSTA, B. G. et al. Fault classification on transmission lines using knn-dtw. In: Computational Science and Its Applications – ICCSA 2017: 17th International Conference, Trieste, Italy, July 3-6, 2017, Proceedings, Part I. Cham: Springer International Publishing, 2017. p. 174–187. ISBN 978-3-319-62392-4. Disponível em: <https://doi.org/10.1007/978-3-319-62392-4_13>.

1.6 Estrutura do trabalho

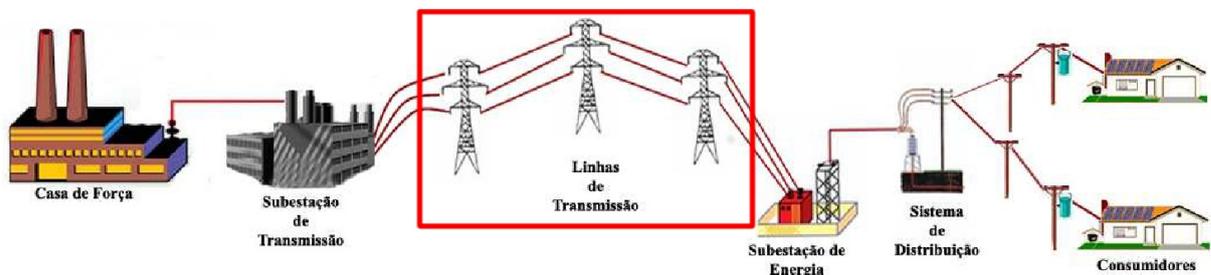
Esta dissertação está dividida em cinco capítulos, organizados da seguinte forma: o primeiro capítulo já discutido, apresenta a introdução, motivação, descrição do problema, a metodologia com seus objetivos, trabalhos relacionados, contribuições do trabalho e a estrutura da pesquisa. No Capítulo 2 é apresentada a fundamentação teórica, mais especificamente os conceitos relacionados a classificação de faltas em linhas de transmissão e os classificadores de sequências utilizados. O Capítulo 3 descreve a base de dados detalhando as características encontradas atualmente, bem como a descrição dos *front ends* adotados. O Capítulo 4 apresenta a configuração dos experimentos realizados e os resultados obtidos. Finalmente, no Capítulo 5 é apresentado as considerações finais e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Introdução

Pesquisas recentes revelam que não há SEP sem ocorrências de distúrbios, mesmo que sejam construídos com as melhores estruturas físicas usadas atualmente (YADAV; DASH, 2015). Dentre os componentes de um SEP, as linhas de transmissão (vide figura 1) são as maiores em termos de extensão, logo estão mais expostas e vulneráveis à oscilações indesejáveis. São vários os distúrbios que podem ocorrer neste componente do sistema elétrico, dentre os quais, destacam-se os curto-circuitos, objeto de estudo desta dissertação.

Figura 1 – Representação do Sistema Elétrico de Potência (SEP)



Fonte: (MORAIS et al., 2010)

Para compreender a complexidade de um sistema de classificação de faltas que representam o curto-circuito em uma linha de transmissão, é necessário entender a principal característica dos dados a serem analisados. Neste sentido, Os dados coletados por aplicações de monitoramento, irão compor uma base de dados, armazenando os valores de interesse do comportamento das formas de onda de tensão e corrente em relação ao tempo.

A maioria dos SEP apresentam três fases (A, B e C) para transmissão de energia elétrica, essas fases devem ser bem distintas, caso ocorra o contato entre elas, temos o curto-circuito. Levando em consideração o terra (T), a combinação de faltas do tipo curto-circuito, pode chegar a no máximo 11 tipos distintos. Contudo, neste trabalho são considerados apenas 10 tipos de curto-circuito, isto porque as faltas ABC e ABCT geralmente não são distinguíveis, já que em sistemas balanceados, como os utilizados nas simulações da base de dados UFPAFaults, não há fluxo de corrente através do terra (MORAIS et al., 2010).

A classificação de faltas em linhas de transmissão, corresponde a um problema de classificação especial, onde as formas de onda de tensão e corrente representam sequências (séries temporais) de tamanho variado, isto é, os dados de entrada são representados por uma matriz de tamanho variável. Assim, a concepção de um modelo de classificação que trate essa matriz de entrada pode ser realizado de várias formas. Por isso, as próximas seções têm como objetivo fornecer uma notação precisa e elucidativa sobre possíveis alternativas.

2.2 Classificação de seqüências representando curto-circuitos em linhas de transmissão

Os sistemas de classificação de seqüências representando curto-circuitos em linhas de transmissão podem ocorrer de duas formas: *on-line* e *off-line*.

Os sistemas de classificação *on-line*, dispõe de um curto segmento do sinal (quadro - *frame*) para classificar as faltas. Geralmente, esses sistemas adotam a classificação quadro a quadro e lidam com vetores de tamanho fixo, condição que permite a utilização de algoritmos convencionais tais como rede neurais. Estes sistemas são geralmente utilizados a nível de proteção de relés, onde a tomada de decisão deve ser automática e em um curto intervalo de tempo. Assim, os sistemas *on-line* tentam resolver problemas que podem ser tratados como problemas de classificação convencional, onde uma decisão deve ser tomada para cada 33 amostras de sinal, por exemplo (HOMCI et al., 2016).

A classificação pós-falta ou *off-line*, lida geralmente com uma quantidade de dados superior a classificação *on-line* e a matriz de entrada possui dimensão variável, onde ao contrário da classificação *on-line*, a classificação pós-falta é realizada a partir de uma seqüência de quadros. Por este motivo os sistemas de classificação pós-falta buscam resolver problemas que podem ser tratados como problemas de classificação de seqüências (MORAIS et al., 2010).

Na classificação de faltas em linhas de transmissão, representada por uma seqüência, geralmente encontrada em um sistema elétrico trifásico, pode-se considerar que cada falta é uma série temporal multivariada de duração variável. A n -ésima falta X_n pode ser representada por uma matriz $Q \times N$. A coluna x_t de $X_n, t = 1, \dots, T_n$, é uma amostra multidimensional representada por um vetor de Q elementos, onde Q é o número de sinais e T_n é o número de amostras da falta. Este trabalho adota $Q = 6$ (formas de onda de tensão e corrente das fases A, B e C) nos experimentos. Como o número de amostras multivariadas depende de n , um classificador convencional não é viável, sendo este o motivo principal para se utilizar o algoritmo DTW como medida de similaridade do classificador convencional KNN.

Figura 2 – Processo de classificação direta de seqüências



Fonte: (HOMCI et al., 2016)

Outra alternativa é classificação de seqüências baseada em quadros (FBSC) apresentada inicialmente no trabalho de (MORAIS et al., 2010) e que será discutida brevemente na próxima seção.

2.3 Classificação de seqüências baseados em quadros

Na arquitetura FBSC as seqüências de entrada são segmentadas em vetores de tamanho fixo denominados de quadros (*frames*). Esta arquitetura utiliza um classificador convencional F como base (rede neural, etc.) para processar cada quadro e obter *scores* $y = \{f_1(z), \dots, f_y(n)\}$ para cada classe. Os *scores* de todos os N quadros são contabilizados para se obter uma decisão final, isto é, verifica-se qual a classe mais frequente entre os quadros classificados. Este processo é denominado na arquitetura de *Voting*.

Um classificador convencional F é um mapeamento $f : \mathbb{R}^K \rightarrow \{1, \dots, Y\}$, onde K é a dimensão do vetor de entrada $z \in \mathbb{R}^K$ e o rótulo $y \in \{1, \dots, Y\}$ é a classe. É utilizado um conjunto de treinamento $T = \{(z_1, y_1), \dots, (z_v, y_v)\}$ contendo V exemplos de (z, y) , para treinar um classificador convencional.

Figura 3 – Fluxo de processamento da arquitetura FBSC. A saída do classificador de seqüência $G(Z)$ depende das decisões do classificador $F(z_n)$



Fonte: (HOMCI et al., 2016)

O processo de classificação da arquitetura FBSC, ocorre após a execução do classificador F , seguindo uma seqüência iniciada com a submissão das amostras de tamanho variável, a um extrator de parâmetros denominado de *front end*, que converte uma matriz X representando as faltas em uma matriz Z com dimensão $K \times N_n$, onde K é o número de parâmetros e N_n o número de vetores de parâmetros do n -ésimo exemplo. Esta arquitetura executa a classificação na matriz Z e não em X . A matriz X é composta pela concatenação das amostras originais e organizadas em uma matriz com dimensões $Q \times L$, onde L é o tamanho do quadro e sua concatenação é $\hat{Z} = [F_1, \dots, F_{N_n}]$, gerando uma matriz com dimensão $Q \times L_n$, e N é número de quadros. No caso de sobreposição é considerado um deslocamento S (quantidade de amostras entre o início de dois quadro consecutivos), podendo ser menor que o tamanho da janela. Uma falta X_n é representado a baixo pelo número de quadros N_n

$$N_n = 1 + \lfloor (T_n - L) / S \rfloor \quad (2.1)$$

e a função *floor* é representada por $\lfloor - \rfloor$. Quando o deslocamento é igual ao tamanho do quadro, $S = L$ (não tem sobreposição) e uma concatenação de amostras representam um quadro onde as matrizes $X = \hat{Z}$ coincidem.

A taxa de erro da classificação é um parâmetro utilizado para avaliar o desempenho de um classificador de seqüência $G(Z)$

$$E_s = \frac{1}{R} \sum_{r=1}^R I(G(Z_r) \neq y_r) \quad (2.2)$$

I é a função indicador, que é 1 caso o argumento seja verdadeiro e 0 caso contrário. R representa a sequência de teste. No caso da FBSC, E_s depende diretamente do desempenho do classificador F .

Os classificadores convencionais utilizados são todos pertencentes ao software de mineração de dados WEKA (*Waikato Environment for Knowledge Analysis*). A próxima seção discutirá brevemente o software WEKA, utilizado nesse trabalho. As seções seguintes listarão os principais classificadores utilizados. Como cada um desses classificadores poderia render material para uma dissertação inteira, ao invés de discuti-los em profundidade, busca-se prioritariamente ilustrar como os mesmos são usados no WEKA.

2.4 Weka

O software WEKA ¹ é um pacote desenvolvido em Java (HALL EIBE FRANK, 2009). Desde 1993, vem sendo utilizado nas principais plataformas e tornou-se referência na comunidade científica como um sistema de mineração de dados. O software é composto por um conjunto de implementações de algoritmo com diversas técnicas de aprendizado de máquina, incluindo algoritmos de regressão, classificação, agrupamento, regras de associação, seleção de parâmetros, armazenamento em cluster e visualização.

WEKA é uma ferramenta *open source* disponível sob a licença GPL - *General Public License* (Licença Pública Geral) e atualmente encontra-se na versão 3.8 e está organizado em 4 módulos. O "*Explore*" é um módulo gráfico que permite a navegação no sistema de arquivo local e tem como opção padrão arquivos no formato ARFF. Este módulo possui interface gráfica que permite rodar os algoritmos suportado pelo WEKA. "*Experimenter*" é o módulo utilizado para fazer testes estatísticos, avaliando diferentes algoritmos simultaneamente, através de interface gráfica. O módulo "*Knowledge Flow*" permite a execução de experimentos orientado a *Data Flows*, utiliza canvas onde os componentes do WEKA formam um fluxo de informação, iniciando com os dados, em seguida os filtros, classificadores, avaliação e visualização. O quarto e ultimo módulo "*Simple Command Line Interface*" utiliza linhas de comando para interagir com o WEKA (HALL EIBE FRANK, 2009).

2.5 Classificadores convencionais

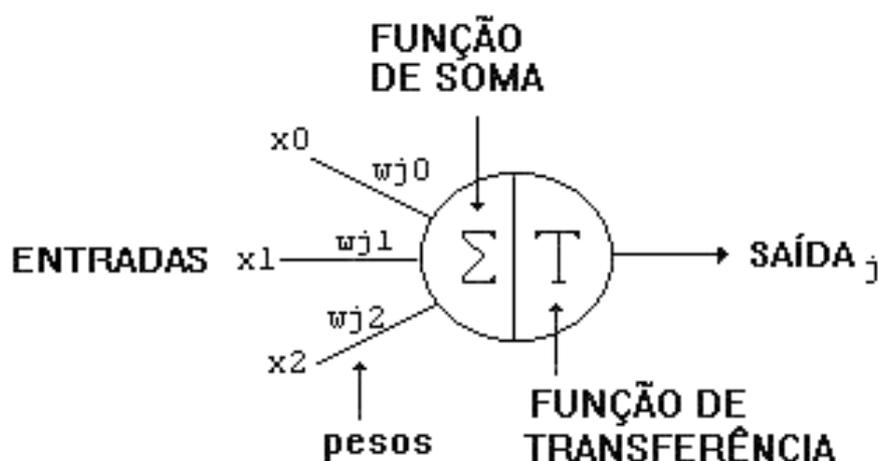
O pacote WEKA disponibiliza diversos algoritmo de aprendizado de máquina, porém na arquitetura FBSC apenas alguns dos principais algoritmos foram utilizados para a classificação de faltas, tais como redes neurais artificiais multicamadas treinadas com algoritmo *backpropagation*, árvore de decisão implementada no WEKA pelo algoritmo J4.8, máquinas de vetores de suporte e *K-nearest neighbor*. Nas próximas subseções serão discutido tais classificadores.

¹ disponível em <http://www.cs.waikato.ac.nz/ml/weka/>

2.5.1 Redes neurais artificiais

Redes neurais artificiais (ANN - *Artificial Neural Network*) são compostos por unidade simples de processamento, denominado de neurônios artificiais que fazem parte de um sistema paralelo distribuído e calculam uma função matemática geralmente não linear (HAYKIN, 2001). Esses neurônios são interligados por intermédio dos chamados pesos sinápticos. As ligações entre as unidades de processamento da rede, são responsáveis pelo o comportamento inteligente da rede neural artificial. A figura 4 exibe a estrutura de um neurônio artificial.

Figura 4 – Estrutura de um neurônio artificial



Fonte: (MORAIS et al., 2010)

Os pesos dos neurônios são ajustados de acordo com função de saída. Mais especificamente, os pesos são corrigidos de modo que se produza a saída desejada diante da respectiva entrada. O cálculo mais utilizado para este fim é a Regra Delta (HAYKIN, 2001).

Em geral o aprendizado da rede neural se dá em função de como os pesos são modificados. Os algoritmos mais conhecidos utilizam aprendizado supervisionado e não supervisionado. No aprendizado supervisionado, a entrada e saída dos dados são conhecidos, este aprendizado ocorre com os ajustes dos pesos, até que os padrões de saída gerados tenham um valor desejado ou próximo do desejado. No aprendizado não supervisionado é construído o aprendizado em função de algumas propriedades do conjunto de dados serem determinadas pela rede neural.

Todos os pares de entrada e saída do conjunto de treinamento do processo de aprendizado, são denominados de iteração ou época. Os ajustes dos pesos numa iteração pode ocorrer no modo *standard* ou em *batch*. A cada apresentação de um conjunto de treino à rede neural é estimado o erro denominado de modo *standard*, enquanto que o erro *batch* é estimado uma média após todos os conjunto de treino serem conhecidos pela rede neural.

O perceptron de múltiplas camadas (MLP) é um modelo de aprendizagem de rede neural, que consiste em um conjunto de nós fontes que compõem a cada de entrada, pode ser encontrada uma ou mais camadas escondidas e uma camada de saída. A forma mais simples de

classificar padrões utilizando redes neurais é com perceptron múltiplas camadas MLP, pois é uma generalização do modelo perceptron.

A dimensão do espaço observado determina a quantidade de neurônios de entrada, enquanto que o número de neurônios de saída é determinado pela dimensionalidade requerida pela resposta. No exemplo da classificação de faltas em linhas de transmissão, o número de sinais de entrada determina a quantidade de neurônios de entrada, enquanto que o número de neurônios de saída pode ter duas representações, uma com 10 neurônios que representam as classes adotadas neste trabalho ou com 4 neurônios de saída, representando as fases A, B, C e T, onde uma falta *AB* pode ser representada por uma resposta da rede como 1100. Considerando que o resultado de uma MLP é determinado pelos seguintes aspectos: número de neurônios da camada escondida, quantidade de camadas escondidas e o algoritmo utilizado para o aprendizado supervisionado.

Neste trabalho foi utilizada a classe *MultilayerPerceptron* do WEKA que implementa a rede neural. Os principais parâmetros dessa classe são:

- -L: Corresponde à taxa de aprendizado utilizada pelo algoritmo *backpropagation*. Este valor deve ser entre 0 e 1 (Padrão é 0.3).
- -M: Taxa de momento para o algoritmo *backpropagation*. Este valor deve ser entre 0 e 1 (Padrão é 0.2).
- -N: Este parâmetro corresponde ao número de épocas para treinamento da rede. O Padrão é 500.
- -H: Corresponde à quantidade de camadas ocultas que podem ser criadas na rede. Por exemplo -H 3, 2 cria duas camadas intermediárias com 3 e 2 neurônios respectivamente. Outra forma de representar pode ser através do uso de letras: a opção *a* corresponde a $(\text{números de atributos} + \text{número de classes})/2$; as outras opções são: *i* (número de atributos), *o* (número de classes) e, *t* (números de atributos + número de classes).

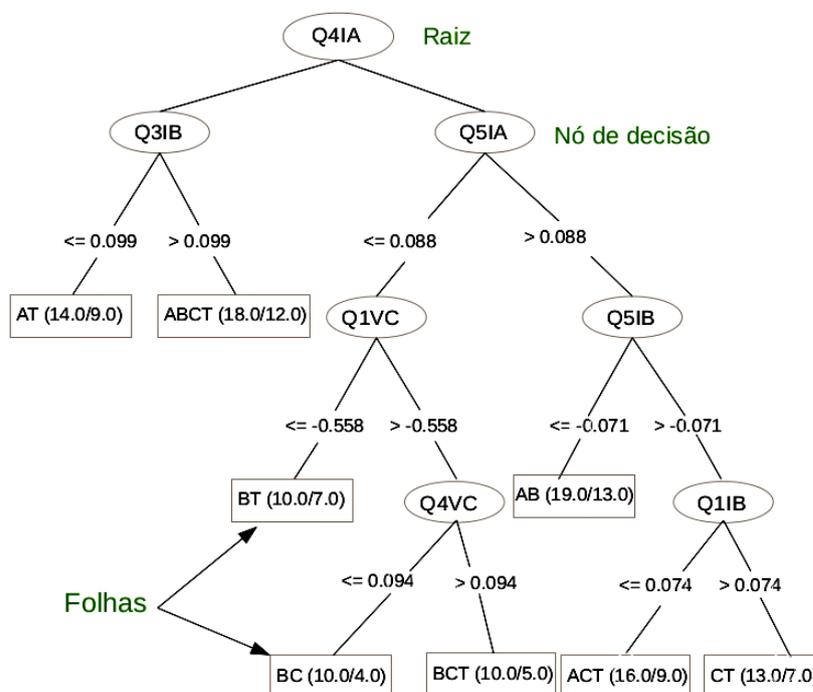
2.5.2 Árvores de decisão

Uma árvore de decisão (figura 5) é constituída de nós e ligações (ramos). Estes nós podem ser classificados em nós internos e externos. Os nós internos apresentam regras que serão submetidas para a tomada de decisão na escolha de um nó filho. A hierarquia presente na organização da árvore, permite que sempre o nó pai tomará a decisão, desde o início com submissão a regra presente na raiz. Seguindo o fluxo gerado pelas decisões dos nós internos, é possível alcançar o nó mais externo da árvore, chamado de folhas ou terminal, onde estão presente as classes de interesse do sistema.

Para cada observação da base de dados, há uma representação na árvore que corresponde apenas um único caminho que vai da raiz a folha. As regras de classificação são compostas por

um antecedente (pré-condição) um conseqüente (conclusão). Os atributos preditivos definem as regras para seleção do antecedente, enquanto que as classes definem os conseqüentes. Na figura 5 temos o seguinte exemplo de regras: Se $Q4IA \leq -0.123$ and $Q3IB$ então é a classe AT.

Figura 5 – Exemplo de uma árvore de decisão.



Fonte: Próprio autor

Para a construção de uma árvore de decisão as escolhas dos atributos (parâmetros), é a principal estratégia para determinar a classe a qual uma *instance* pertence. No exemplo da figura 5 a raiz da árvore 4QIA, foi avaliado pelo algoritmo como parâmetro mais importante para classificar o tipo de falta (curto-circuito) em linhas de transmissão.

A classe utilizada no WEKA que implementada a árvore de decisão foi a J4.8. A árvore J4.8 constrói um modelo de árvore de decisão baseado em um conjunto de dados de treinamento e usa esse modelo para classificar as instâncias do conjunto de teste. A seguir são apresentados os principais parâmetros desse classificador implementado no WEKA.

- -U: não utiliza poda da árvore e a mesma cresce até seu tamanho máximo
- -C: Corresponde ao fator limiar de confiança de poda. O padrão é 0.25.
- -M: indica o número mínimo de exemplos por folha (padrão 2).
- -R: usa o método de erro de redução da poda (QUINLAN, 1993).
- -N: indica o número de campos para o erro de redução de poda. Um campo é usado como conjunto de poda (padrão 3).

2.5.3 Máquinas de vetores de suporte

As máquinas de vetores de suporte (*Support Vector Machine* - SVM) utiliza a teoria de aprendizado estatístico, como técnica de aprendizado de máquina. O classificador SVM consiste em encontrar a máxima separação em um espaço de características, para localizar um hiperplano que tem uma função em sua superfície que determina a separação máxima de uma classe da outra.

Na literatura encontra-se diversas possibilidades da função *kernel* ou núcleo, um ponto importante para execução de aprendizagem por vetor de suporte. Aplicações para o reconhecimento de padrões, apresentam as seguintes possibilidades de *kernel* SVM: linear, polinomial, sigmóide e funções de base radial (RBF - *Radial Basis Function*).

A função de estimação ϕ a seguir, pode caracterizar as SVMs e outros métodos *kernel*.

$$\frac{1}{V} \sum_{v=1}^V L(\phi(x_v), y_v) + \lambda \|\phi\|_{H_\kappa}^2, \quad (2.3)$$

onde:

- H_κ corresponde ao espaço euclidiano gerado pelo *kernel* κ ;
- $\phi = h + b$, onde h corresponde ao produto do vetor peso (ω) pelo vetor de suporte (x_v) com $h \in H_\kappa$;
- b corresponde ao bias, $b \in \mathbb{R}$;
- $L(\phi(x_v), y_v)$ corresponde a função perda (risco fundamental);
- λ corresponde a autovalores;
- V corresponde ao número de exemplos de treino.

Neste trabalho, a função *kernel* utilizada é o RBF, e os principais parâmetros utilizados foram o $-C$ ($C > 0$), parâmetros de penalidade do termo de erro, e o $-G$, largura da função *kernel* RBF.

2.5.4 K-vizinhos mais próximos

Diferente dos outros classificadores, o KNN (*K-Nearest Neighbors*) ou K-vizinhos mais próximo, não constrói um modelo de classificação com a base de treino e sim utiliza os dados de treinamento como modelo de classificação para reconhecimento de um novo padrão, este algoritmo executa uma varredura nos dados e compara com todas as amostras de treino para encontrar o novo padrão a ser classificado.

Supondo que $Z = (z_1, \dots, z_K)$ é uma sequência a ser o padrão reconhecido pelo KNN, então será calculada a distância de similaridade em relação a todos os exemplos do conjunto de

treinamento V , para descobrir os vizinhos mais próximos. Em seguida será verificado qual classe aparece com maior frequência, entre os K vizinhos encontrados, o padrão z será classificado com a classe y mais frequentes dentre os K exemplos encontrados.

O cálculo da medida de similaridade entre duas *instances* é convencionalmente calculado com a distância euclidiana, tal medida calcula a raiz quadrada da norma do vetor diferença entre os vetores z e v :

$$d(z, v) = \sqrt{\sum_{i=1}^K (z_i - v_i)^2} \quad (2.4)$$

No *software* WEKA o KNN é implementado pela classe IBK e seus principais parâmetros são:

- -N: número de centros ou K (vizinhos).
- -S: esta opção gera aleatoriamente os centros.

2.6 Classificação de sequências baseada no classificador KNN-DTW

O algoritmo de Alinhamento Temporal Dinâmico (DTW) é uma medida de similaridade baseada no conceito de programação dinâmica que permite comparar sequências de durações variáveis. É um algoritmo que se popularizou em aplicações de reconhecimento de voz, sendo atualmente bastante utilizado em áreas que lidam com séries temporais (GIORGINO, 2009).

Antes de entender o conceito de DTW é importante um entendimento da definição de programação dinâmica. A programação dinâmica é um método de programação aplicável em situações nas quais não é fácil chegar a uma sequência ótima de decisões sem testar todas as sequências possíveis para então escolher a melhor. A idéia básica da programação dinâmica é construir por etapas uma resposta ótima combinando respostas já obtidas em partes menores (YURTMAN; BARSHAN, 2013).

Nesse sentido, o alinhamento usando DTW entre duas séries temporais de faltas $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_p)$ e $\hat{\mathbf{Z}} = (\hat{\mathbf{z}}_1, \dots, \hat{\mathbf{z}}_q)$, consiste em gerar uma matriz \mathbf{D} de dimensão $p \times q$ cujo elemento (i, j) contém a distância euclidiana $d(\mathbf{z}_i, \hat{\mathbf{z}}_j)$ (outras medidas de similaridade podem ser adotadas) entre os vetores \mathbf{z}_i e $\hat{\mathbf{z}}_j$. Um percurso de ajuste W , é um conjunto de elementos contínuos da matriz que define um mapeamento entre os vetores de \mathbf{Z} e $\hat{\mathbf{Z}}$. O k -ésimo elemento de W é definido como $w_k = (i, j)_k$. Portanto

$$W = w_1, w_2, \dots, w_{|W|}, \max(q, p) \leq |W| < p + q - 1, \quad (2.5)$$

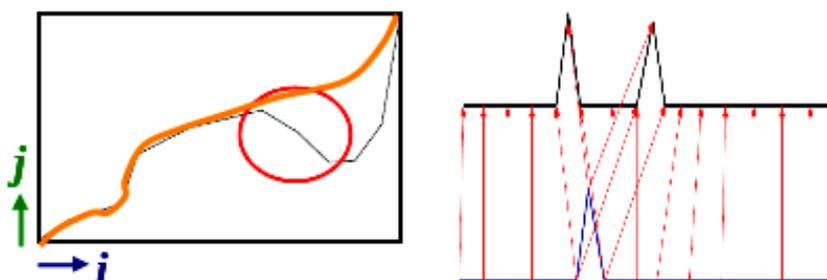
onde $|W|$ é a cardinalidade (número de elementos) de W . O elemento $\mathbf{D}(w_k)$ corresponde à distância euclidiana respectiva ao elemento w_k da matriz.

O DTW tem objetivo de encontrar o menor caminho entre duas sequências, conhecer todos os caminhos não é a solução ótima, visto que a quantidade de caminhos possíveis na matriz formada pelas sequências \mathbf{Z} e $\hat{\mathbf{Z}}$ é de ordem exponencial, por esse motivo é necessário reduzir o espaço de busca utilizando algumas estratégias. As estratégias adotadas são: monotocidade, continuidade, condição de contorno, alinhamento de janela e restrições de inclinação.

- Monotocidade

A monotocidade garante que não se repita o mesmo ponto, pois permanece ou incrementa os valores de i e j onde, $i_{s_1} \leq i_s$ e $j_{s_1} \leq j_s$, logo não retorna a mesma posição da matriz.

Figura 6 – Monotocidade

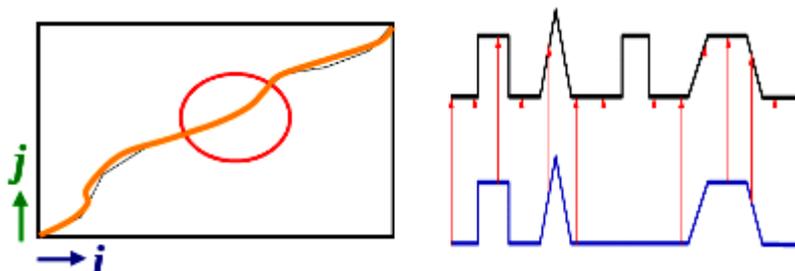


Fonte: adaptado de <http://www.psb.ugent.be/cbd/papers/gentxwarper/DTWalgorit>

- Continuidade

Na continuidade não há saltos na sequência do caminho, a condição $i_s, s-1 \leq 1$ e $j_s - i_{s-1} \leq 1$, garante que o alinhamento não omita uma característica importante.

Figura 7 – Continuidade

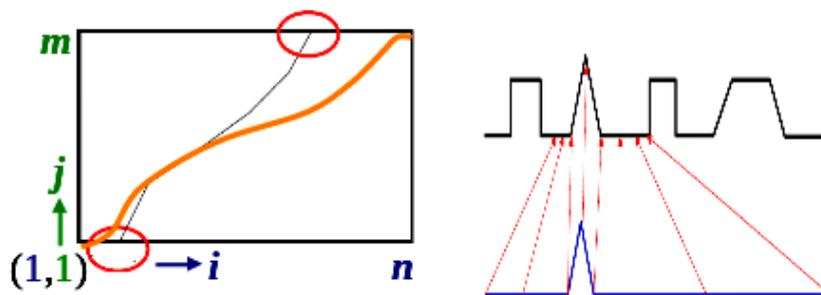


Fonte: adaptado de <http://www.psb.ugent.be/cbd/papers/gentxwarper/DTWalgorit>

- Condição de contorno

Na condição de contorno o caminho começa no canto inferior esquerdo e termina no canto superior direito, a condição $i_1 = 1, i_k = n$ e $j_1 = 1, j_k = m$, garante que o alinhamento não considera uma sequência parcial.

Figura 8 – Condição de contorno

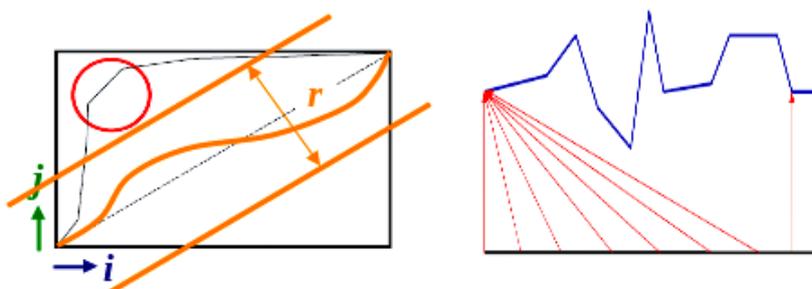


Fonte: adaptado de <http://www.psb.ugent.be/cbd/papers/gentxwarper/DTWalgorit>

- Alinhamento de janela

O alinhamento de janela limita o caminho evitando que exceda a largura da janela, garantindo o alinhamento próximo a diagonal, a condição $|i_s - j_s| \leq r$, onde $r > 0$ é o tamanho da janela, garante que caminho não permita características diferentes e nem estrutura similar.

Figura 9 – Alinhamento de janela

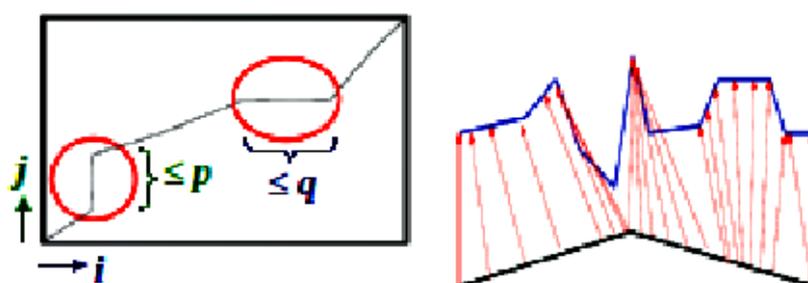


Fonte: adaptado de <http://www.psb.ugent.be/cbd/papers/gentxwarper/DTWalgorit>

- Restrição de inclinação

A restrição de inclinação evita que o caminho seja muito inclinado ou muito plano, logo impede que passos curtos coincidam com os longos, a condição é expressa por uma razão de $S = p/q \in [0, \infty]$ em que p é o número de passos desejados na mesma direção (vertical ou horizontal) e q é o tempo na direção diagonal (SILVA; BATISTA,). Restringe com a seguinte condição,

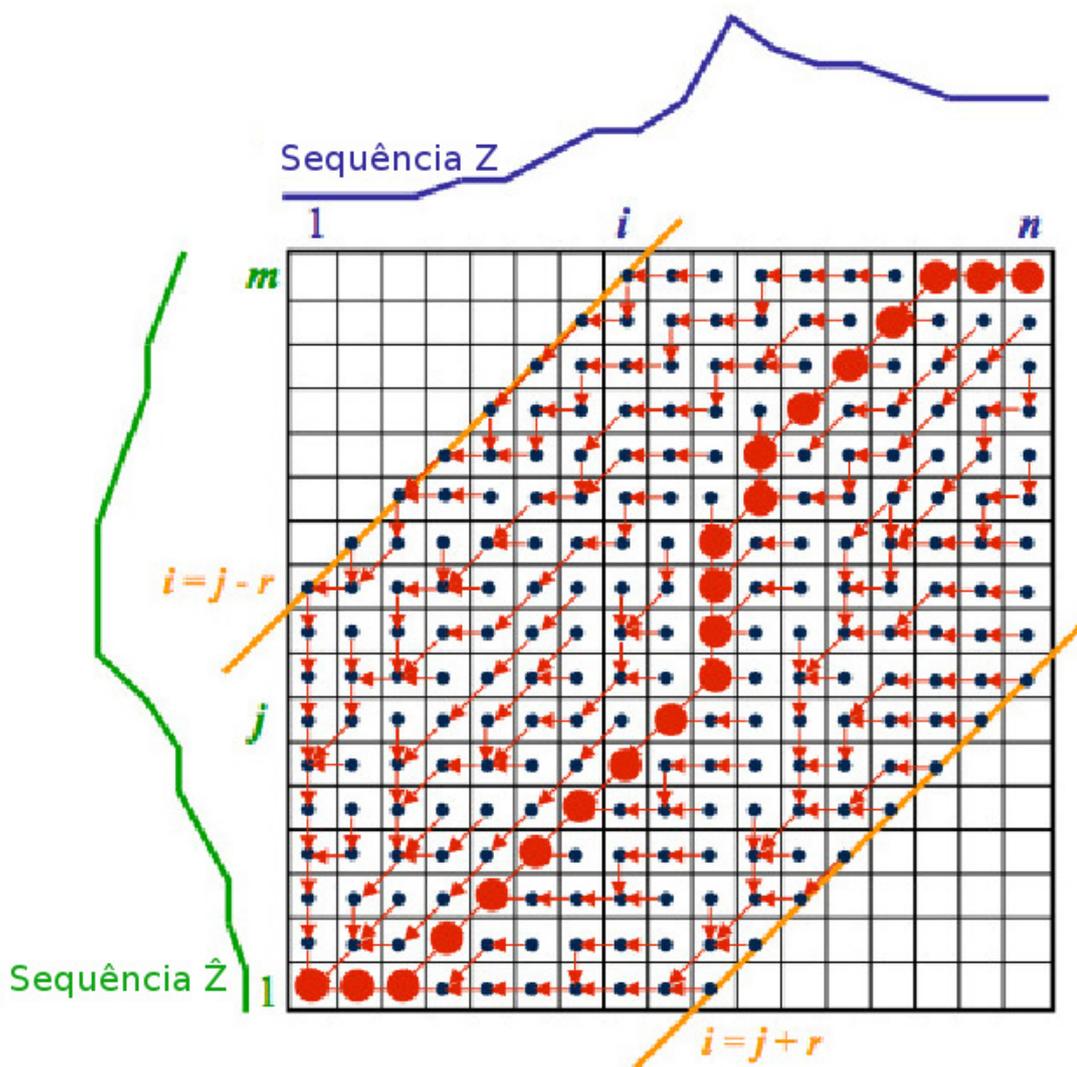
Figura 10 – Restrição de inclinação



Fonte: adaptado de <http://www.psb.ugent.be/cbd/papers/gentxwarper/DTWalgorit>

A figura 11 é uma representação clássica do resultado final do algoritmo DTW e exhibe as sequências Z e \hat{Z} , a matriz de alinhamento, as restrições e a sequência de interesse encontrada.

Figura 11 – Sequência encontrada após o cálculo DTW



Fonte: <http://www.psb.ugent.be/cbd/papers/gentxwarper/DTWalgorit>

Dos muitos percursos de ajuste que satisfazem as condições acima, o percurso escolhido é aquele que minimiza o custo da deformação:

$$DTW(\mathbf{Z}, \hat{\mathbf{Z}}) = \min_W \sum_{k=1}^{|W|} \frac{\mathbf{D}(w_k)}{|W|} \quad (2.6)$$

onde $|W|$ é usado para compensar o fato de que o percurso de ajuste tenha diferentes tamanhos.

Este percurso pode ser encontrado de modo eficiente usando a programação dinâmica para avaliar a recursão sucessiva que define a distância acumulativa $\gamma(i, j)$. Um possível exemplo de cálculo de $\gamma(i, j)$, o qual foi usado nesse trabalho, é (vide (GIORGINO, 2009) para outras opções):

$$\gamma(i, j) = \mathbf{D}(i, j) + \min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\}. \quad (2.7)$$

Nota-se que tal cálculo é baseado na distância $\mathbf{D}(i, j)$ encontrada na célula corrente e o mínimo das distâncias acumuladas dos elementos adjacentes.

Como mencionado anteriormente, o DTW foi usado como medida de similaridade do algoritmo KNN aplicado à classificação de sequências, visto que utilizar a distância euclidiana como medida de similaridade, na classificação de sequências, não é possível em muitos casos ou mesmo o mais recomendável (KEOGH; RATANAMAHATANA, 2005; XI et al., 2006). Neste caso, o classificador KNN para o módulo de classificação de sequências é chamado de KNN-DTW (XI et al., 2006).

3 METODOLOGIA PROPOSTA

3.1 Base de dados

O acesso na literatura a resultados utilizando base de dados com eventos de QEE não é uma tarefa trivial, visto que a maioria delas são proprietárias e fatores como tempo, custo e confidencialidade dos dados, fazem com que as bases não sejam disponibilizadas e a reprodução de resultados seja uma tarefa complicada ou mesmo impossível. Inclui-se nesse contexto, a disponibilização de bases rotuladas para atender aplicações de mineração de dados, como por exemplo a classificação de faltas em linhas de transmissão foco deste trabalho (MORAIS et al., 2010).

Este trabalho utiliza uma base de dados de domínio público, e rotulada com faltas do tipo curto-circuito em linhas de transmissão chamada UFPAFaults. Esta base de dados foi desenvolvida e disponibilizada pelo grupo de pesquisa do Laboratório de Processamento de Sinais (LaPS) da Universidade Federal do Pará (UFPA). Ao longo do tempo a base foi sendo incrementada e a versão mais atual possui 27500 simulações, organizadas em cinco conjuntos de 100, 200, ..., 1000 faltas cada. Todas as simulações, realizadas no intervalo de 1 segundo, são formas de onda de tensão e corrente representando faltas do tipo curto-circuito em linhas de transmissão. Considera-se que, as mesmas podem ocorrer com a mesma probabilidade em uma linha ou em qualquer uma das linhas do circuito considerado.

A base de dados UFPAFaults é gerada utilizando dois softwares: ATP (*Alternative Transient Program*) (SCOTT; LIU, 1992) e AmazonTP (YOMARA et al., 2006). ATP é um simulador bastante conhecido na área de sistemas de potência para execução de análises e simulações digitais de eventos em SEP. Basicamente, ele gera um modelo do circuito elétrico a ser simulado, apresentando simulação de tensão, corrente, potência e energia. O papel do AmazonTP é invocar o ATP repetidas vezes, manipulando os modelos gerados conforme a necessidade. Para este trabalho, os modelos são manipulados através do fechamento de chaves do circuito, a fim de se obter faltas simuladas.

As simulações têm duração de 1 segundo e são apresentados em dois tipos de arquivos. O primeiro é um TXT que possui informações gerais sobre todas as simulações, como numeração, tipo de falta, quando a falta começa e termina, entre outros. O segundo arquivo está no formato ASCII, e é o que contém as formas de onda em si, passando o intervalo de tempo e valores de tensão e corrente.

Na figura 12 é possível verificar o cabeçalho e como os dados estão organizados por coluna nos arquivos ASCII. Este arquivo é organizado da seguinte maneira: a primeira coluna indica o intervalo de tempo, as colunas 2, 3 e 4 correspondem aos valores de tensão e as colunas 5, 6 e 7 indicam os valores de corrente.

Em síntese, para 1000 simulações devem existir um arquivo contendo as informações

gerais sobre as faltas de cada simulação e 1000 arquivos com informações das formas de onda, considerando uma frequência de amostragem de 40 KHz. A base utiliza 11 tipos de faltas uniformemente distribuídas: AT, BT, CT, AB, AC, BC, ABC, ABT, ACT, BCT e ABCT. Vale destacar que ABC e ABCT são consideradas a mesma classe neste trabalho.

Figura 12 – Arquivo ASCII representando as informações associadas disponibilizada na base UF-PAFaults.

25-Feb-08 09.32.38 94 6 12 158 0 10E8.0										
PHASOR SSOMIT LIN001 LIN002 LIN003 LIN004 LIN005 LIN006 LIN007 LIN008										
LIN009 LIN010 LIN011 LIN012 LIN013 LIN014 LIN015 LIN016 LIN017 LIN018 LIN019										
LIN020 LIN021 LIN022 LIN023 LIN024 LIN025 LIN026 LIN027 LIN028 LIN029 LIN030										
LIN031 LIN032 LIN033 LIN034 LIN035 LIN036 LIN037 LIN038 LIN039 LIN040 LIN041										
LIN042 LIN043 LIN044 LIN045 LIN046 LIN047 LIN048 LIN049 LIN050 LIN051 LIN052										
LIN053 LIN054 LIN055 LIN056 LIN057 LIN058 LIN059 LIN060 LIN061 LIN062 LIN063										
LIN064 LIN065 LIN066 LIN067 LIN068 LIN069 LIN070 LIN071 LIN072 LIN073 LIN074										
LIN075 LIN076 LIN077 LIN078 LIN079 LIN080 LIN081 LIN082 LIN083 LIN084 LIN085										
LIN086 LIN087 LIN088 LIN089 LIN090 LIN091 LIN092 LIN093 LIN094 LIN095 LIN096										
LIN097 LIN098 LIN099 LIN100 LIN101 LIN102 NLN001 NLN002 NLN003 NLN004 NLN005										
NLN006 NLN007 NLN008 NLN009 NLN010 NLN011 NLN012 SWT001 SWT002 SWT003 SWT004										
SWT005 SWT006 SWT007 SWT008 SWT009 SWT010 SWT011 SWT012 SWT013 SWT014 SWT015										
SWT016 SWT017 SWT018 SWT019 SWT020 SWT021 SWT022 SWT023 SWT024 SWT025										
SWT026 SWT027 SWT028 SWT029 SWT030 SWT031 SWT032 SWT033 SWT034 SWT035										
SWT036 SWT037 SWT038 SWT039 RUA										
NEC02B NEC02A NEC01C NEC01B NEC01A NEC00C NEC00B NEC00A X0095A X0096B X0096A										
X0038C X0038B X0038A TU230C X0052C TU230B X0052B TU230A X0052A MIDA02 MIDB02										
MIDC02 MIDA01 MIDB01 MIDC01 MIDA00 MIDB00 MIDC00 END_00 NEW00C NEW00B NEW00A										
END_01 NEW01C NEW01B NEW01A END_02 NEW02C NEW02B NEW02A X0063C X0063B X0063A										
TR232C TR232B TR231C TR232A TR231B TR231A AL232C AL232B AL231C AL232A AL231B										
AL231A X0026C X0026B X0026A X0019C X0019B X0019A X0012C X0012B X0012A TU231C										
TU231B TU231A X0078A X0078B X0078C TU132A TU132C TU132B TU133A TU133C TU133B										
TU131A TU131C TU131B TU134A TU134C TU134B										
79 0 78 0 77 0 19 79 18 78										
17 77										
0.0 180246. -18403. -.162E6 -8.359 273.245 -264.89										
.25E-4 179457. -16541. -.163E6 -11.287 274.628 -263.34										
.5E-4 178652. -14677. -.164E6 -14.213 275.988 -261.77										
.75E-4 177832. -12812. -.165E6 -17.139 277.322 -260.18										
.1E-3 176996. -10946. -.166E6 -20.062 278.633 -258.57										
.125E-3 176144. -9079. -.167E6 -22.984 279.918 -256.93										
.15E-3 175277. -7211.1 -.168E6 -25.905 281.178 -255.27										
.175E-3 174394. -5342.5 -.169E6 -28.822 282.414 -253.59										
.2E-3 173495. -3473.5 -.17E6 -31.738 283.624 -251.89										
.225E-3 172581. -1604.2 -.171E6 -34.65 284.81 -250.16										
.25E-3 171652. 265.309 -.172E6 -37.559 285.97 -248.41										
.275E-3 170708. 2134.76 -.173E6 -40.465 287.104 -246.64										

Fonte: Próprio autor

É importante ressaltar que, as amostras disponibilizadas na base passam por um processo de pré-processamento que tem como objetivo aplicar técnicas para permitir que os dados sejam representados de uma forma mais adequada, sem alterar sua organização ao longo do tempo. Este processo faz parte da parametrização da base que ocorre em três etapas: primeiramente realiza-se o pré-processamento que consiste em reamostrar os dados para uma frequência de amostragem que neste caso foi de 2 kHz e normalizar os dados para intervalo entre -1 e 1.

Em seguida, define-se quais os parâmetros que serão utilizados pelos algoritmos de aprendizagem, para isso utilizam-se os chamados *front ends* que vão organizar a matriz de entrada que será submetida ao algoritmos de classificação. A próxima seção discute os *front ends* adotados neste trabalho.

3.2 Front ends

A arquitetura FBSC faz uso de *front ends*, uma fase de pré-processamento que organiza os dados em uma matriz de tamanho fixo para ser processado por um classificador convencional. Alguns *front ends* e a concatenação destes geram um conjunto de dados de alta dimensionalidade, aumentando o custo computacional dos classificadores. Com objetivo de reduzir a alta dimensionalidade dos dados de entrada, pode ser utilizada uma etapa seleção de parâmetros antes do estágio de reconhecimento de padrões como foi feito em (HOMCI et al., 2016). Esta etapa de seleção de parâmetros não foi realizada neste trabalho.

Na arquitetura FBSC o usuário escolhe o *front end*, o classificador e a regra de combinação, isso permite um alto grau de liberdade na concepção do classificador, devendo ser avaliado rigorosamente para evitar conclusões tendenciosas.

O *front end* é responsável por todas as operações de extração de parâmetros que geram as sequências que serão passadas aos algoritmos de classificação. Todos os *front ends* nesse trabalho assumem sequências \mathbf{Z} com $Q = 6$ formas de onda de tensão e corrente. Neste sentido, o presente capítulo tem como objetivo apresentar uma descrição dos *front ends* introduzidos inicialmente em (MORAIS et al., 2010).

3.2.1 Direto da forma de onda - *Raw*

Qualquer amostra que representa uma falta, não contém características suficientes que permita a decisão utilizando um algoritmo de classificação. Por este motivo, um *front end* tem a função de converter as amostras em parâmetros (*features*), gerando uma sequência que permita decisões razoáveis através dos classificadores convencionais.

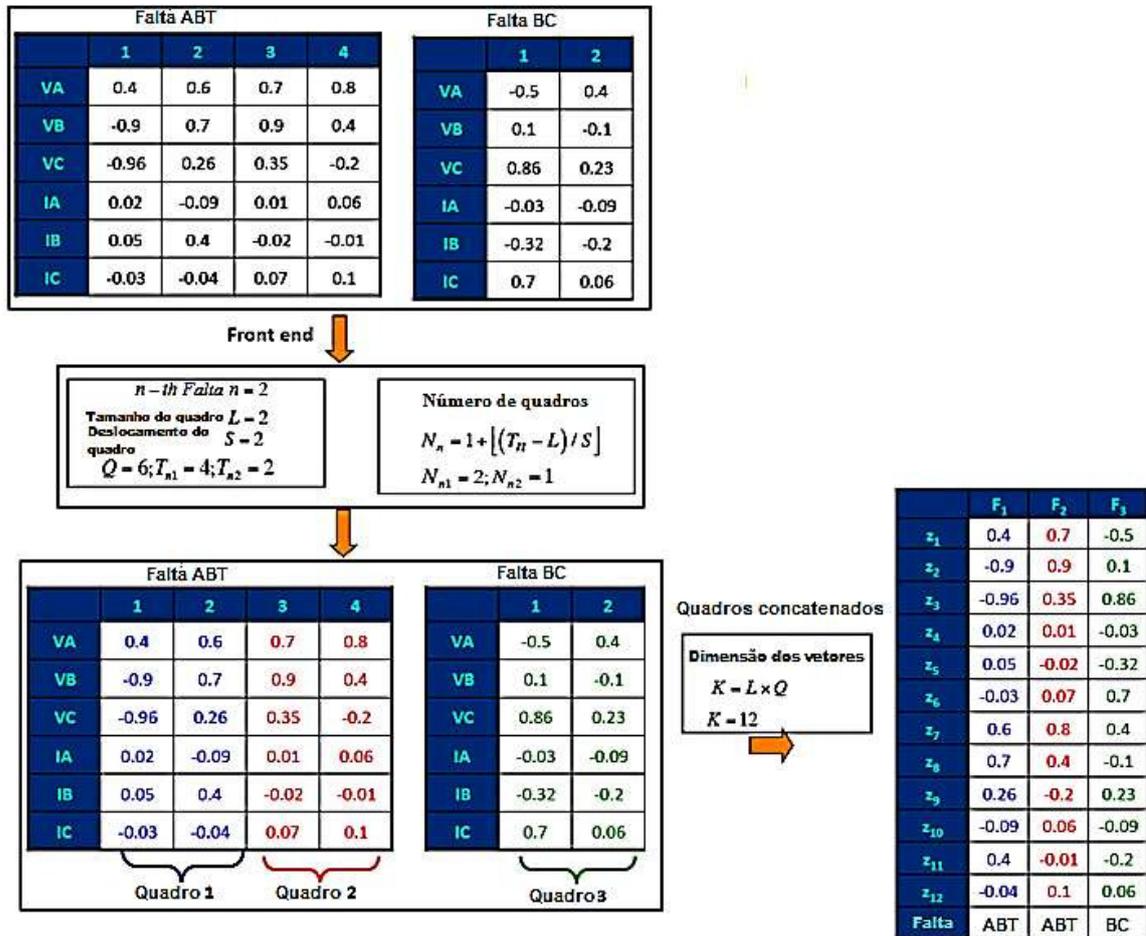
O *front end raw* é o *front end* mais simples, pois seus parâmetros de saída correspondem a valores da amostra original, sem qualquer outro processamento que organize as amostras para uma matriz \mathbf{Z} onde ocorrerá a classificação.

As amostras da matriz \mathbf{Z} são organizadas de acordo com o tamanho da janela (quadro - *frame*) L e o deslocamento do mesmo representado por S . Deve-se ressaltar que se $S = L$ (não há sobreposição) e um quadro é uma *concatenação de amostras*

$$\mathbf{F}_n = [\mathbf{x}_{(1+L(n-1))}, \mathbf{x}_{(2+L(n-1))}, \dots, \mathbf{x}_{(L+L(n-1))}], \quad (3.1)$$

Mais especificamente, se $Q = 6$ (tensões e correntes), um *front end raw* poderia obter quadros \mathbf{F}_n de dimensão 6×5 concatenando para cada amostra central seus quatro vizinhos, dois à esquerda e dois à direita. Neste caso, supondo uma falta com $T_m = 10$ amostras e $L = S = 5$, os valores de K e N_m são respectivamente, 30 e 2. Desta forma, $\hat{\mathbf{Z}}$ e \mathbf{Z} poderiam ter dimensões 6×10 e 30×2 , respectivamente. A figura 13 ilustra a segmentação de uma falta “ABT” (4 amostras) e uma falta “BC” (2 amostras) em vetores de parâmetros com 2 e 1 quadro, respectivamente. Neste exemplo, $L = 2$ e $S = 2$ (não há sobreposição) o que fornece 3 vetores \mathbf{z} , cada um de dimensão $K = 12$.

Figura 13 – Demonstração do *front em raw*, organizando os vetores de padrão z . Neste exemplo, $L = 2$ e $S = 2$ (não há sobreposição) e a matriz obtida pela aplicação do *front end* possui dimensão $K = 12$ com quadros da falta “ABT” e um quadro falta “BC”.



Fonte: Próprio autor

3.2.2 Front end RMS

Várias normas de QEE classificam os fenômenos de acordo com as características, como o tempo em que o valor médio o quadrático (*RMS - Root Mean Square*) da tensão extrapolou um limiar. O RMS é um dos *front ends* mais usados no processo de parametrização de amostras, permitindo uma maior interpretabilidade quando comparado a outros *front ends*, além de permitir uma maior estimativa aproximada da amplitude da frequência fundamental de uma forma de onda (MORAIS et al., 2010).

Este *front end* consiste em calcular o valor RMS janelado para cada uma das formas de onda Q . Considerando o tamanho da janela L e o deslocamento S , o n -ésimo valor RMS $z[n]$, $n = 1, \dots, N$ de uma forma de onda $x[t]$, $t = 1, \dots, T$ é dado por

$$z[n] = \sqrt{\frac{1}{L} \sum_{l=1}^L (x[l + (n-1)S])^2}, \quad (3.2)$$

onde N e T são apresentados na Eq.(2.1).

Onde:

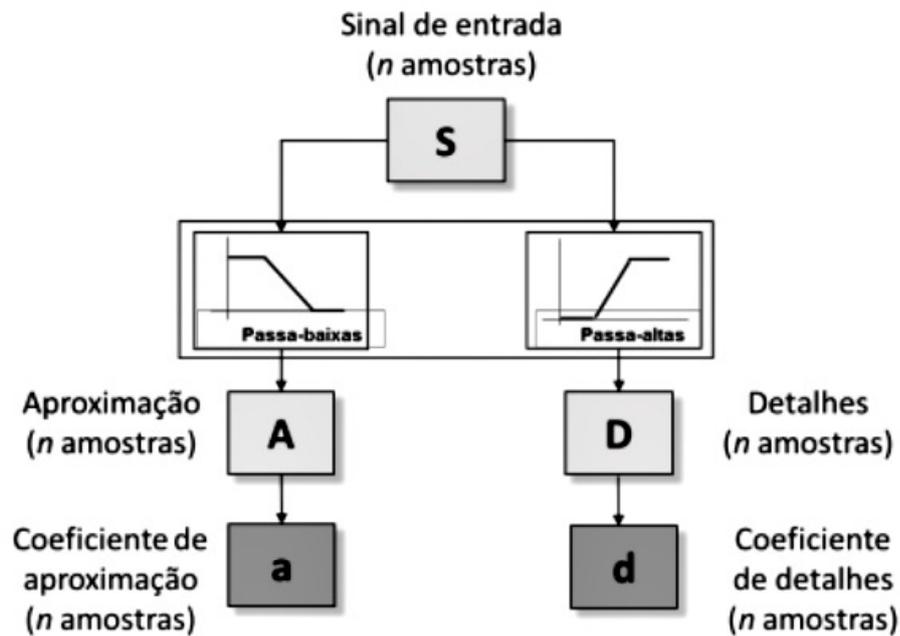
- N é calculado de acordo em equação 2.1
- T é o número de amostras.
- L é o tamanho do quadro.
- S é o deslocamento do quadro.

3.2.3 *Front ends wavelets*

São *front ends* classificados como multi-resolução (um sinal pode ser analisado em diferentes frequências com diferentes resoluções), que utilizam a técnica de transformada *wavelet* para fornecer informações do sinal no domínio do tempo e frequência simultaneamente, sendo bastante utilizados para prover estudos em sistemas elétricos pois, são capazes de identificar características únicas das diferentes faltas localizadas nos vários níveis de decomposição do sinal para o processo de classificação de faltas em linha de transmissão (SILVA et al., 2006; SILVA; SOUZA; BRITO, 2006; VETTERLI; KOVAČEVIĆ, 1995).

O sinal analisado é dividido em componentes de alta escala e baixa escala. As primeiras são chamadas de “aproximações”, já que correspondem ao conteúdo de baixa frequência do sinal. As variações rápidas do sinal são chamadas de “detalhes” para exemplificar, no seu nível mais básico, o processo de filtragem que gera a divisão mencionada pode ser entendido através da ilustração da figura 14, onde são utilizados dois filtros para decompor o sinal S , obtidos através da sua convolução repetidas vezes. O primeiro é um filtro passa-baixa, que representa a geração do conteúdo de baixa frequência ou aqui chamado de coeficiente de aproximação \mathbf{a} ; e um segundo filtro passa-alta com o conteúdo de alta-frequência com os detalhes representados pelos coeficientes de detalhe \mathbf{d} .

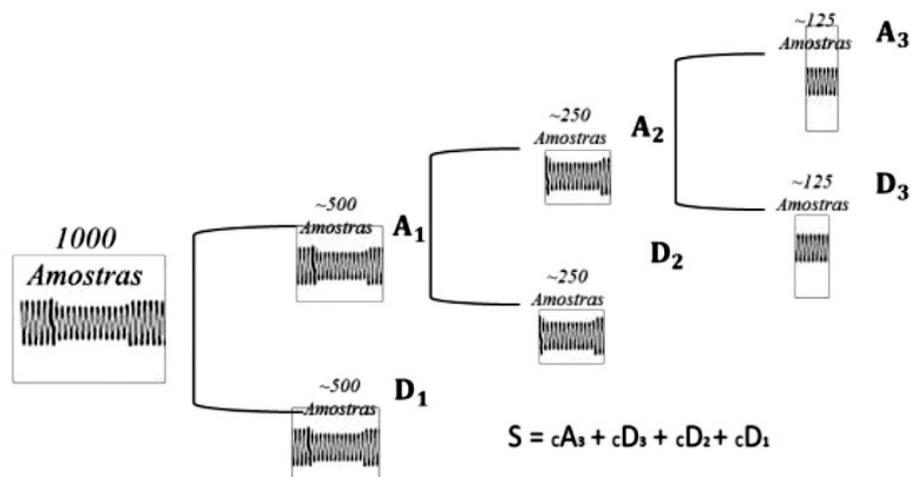
Figura 14 – Filtragem de um estágio para geração de aproximações (A) e detalhes (D) de um sinal (S).



Fonte: Próprio autor

Após a decomposição do sinal, a primeira por si só é então decomposta em aproximações e detalhes de segundo nível, e o processo se repete, como ilustrado na figura 15. Para n níveis, a *wavelet* oferece $n + 1$ possibilidades de decomposição do sinal.

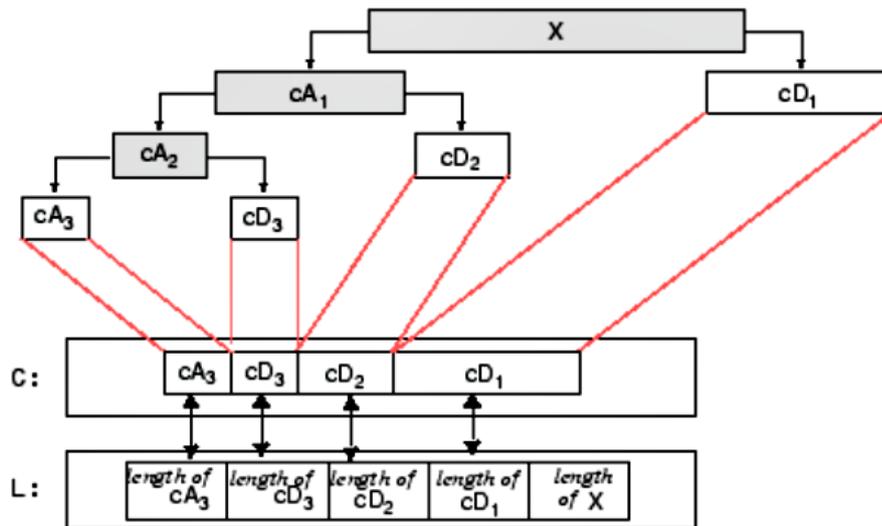
Figura 15 – Exemplo de decomposição de 3 níveis.



Fonte: Próprio autor

A figura 16 ilustra como ocorre o processo de decomposição *wavelet* utilizando a função *wavedec* do MATLAB, para 3 níveis de decomposição. Essa função retorna dois vetores **C** e **L**, onde **C** é um vetor que armazena os coeficientes de decomposição (cD_3 , cD_2 e cD_1) e o último de aproximação (cA_3), e **L** contém os tamanhos de cada coeficiente de **C** e o tamanho do sinal original.

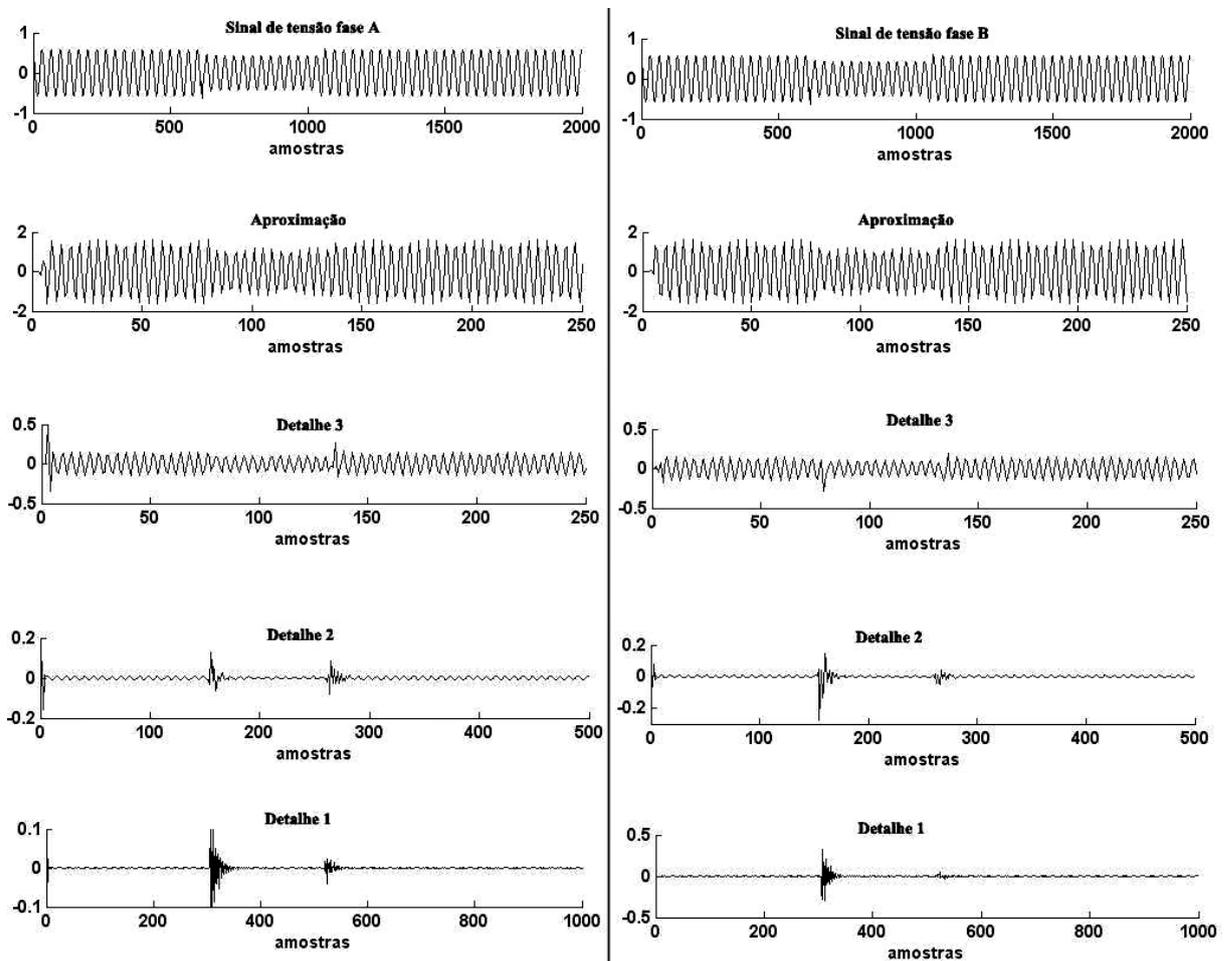
Figura 16 – Decomposição wavelet a partir da função *wavedec* do MATLAB.



Fonte: Próprio autor

Assume-se uma decomposição wavelet diádica, que tem γ estágios de filtragem e decimação (VETTERLI; KOVAČEVIĆ, 1995) e transforma cada uma das formas de onda Q em $\gamma + 1$ formas de onda. Mais especificamente, a q -ésima forma de onda é decomposta em aproximação q e detalhes $^q_{1,2}, \dots, ^q_{\gamma}$, para $q = 1, \dots, Q$. Por simplicidade, a dependência em relação a q será omitida. A figura 17 apresenta um exemplo de aplicação da transformada wavelet aos sinais de tensão das fases A e B de uma falta AB.

Figura 17 – Exemplo de uma decomposição *wavelet* aplicada ao sinais de tensão das fases A e B de uma falta AB simulada no intervalo de 1s. A *wavelet* mãe é uma *Daubechies 4* com três níveis de decomposição ($\gamma = 3$).



Fonte: Próprio autor

Alguns trabalhos na literatura usam somente um dos coeficientes de detalhe (AGUILERA; ORDUNA; RATA, 2006; SAWATPIPAT; TAYJASANANT, 2010; SINGH; PANIGRAHI; MAHESHWARI, 2011). Neste caso, a decomposição *wavelet* discreta no tempo é equivalente a uma operação de filtragem digital simples. Além disso, não compensa realizar a decomposição *wavelet* por completo para depois descartar todos os conjuntos de coeficientes e no final armazenar somente um. Neste caso, é mais eficiente implementar diretamente uma filtragem digital com a banda passante de interesse.

Neste trabalho, o *front end* *wavelet* concatena todos os coeficientes e organiza-os em uma matriz \mathbf{Z} . Este leva em consideração que para $\gamma > 1$ os coeficientes possuem diferentes frequências de amostragem. Para isso, em vez de usar um único L , o usuário especifica um valor L_{\min} para as formas de onda com a menor frequência de amostragem f_s (\mathbf{a} e \mathbf{d}_γ) e um valor maior $L_i = 2^{\gamma-i}L_{\min}$ é automaticamente adotado para outros detalhes $i = 1, \dots, \gamma - 1$. Por exemplo, assumindo uma aplicação *wavelet* com 3 níveis de decomposição ($\gamma = 3$), o tamanho dos quadros para \mathbf{d}_1 e \mathbf{d}_2 são, respectivamente, $4L_{\min}$ e $2L_{\min}$. Um raciocínio similar é aplicado

para o deslocamento do quadro $S_i = 2^{\gamma-i}L_{\min}$, onde S_{\min} é outro parâmetro definido pelo usuário.

Em suma, após aplicação da decomposição *wavelet*, os coeficientes são então organizados em um quadro \mathbf{F} de dimensão $Q \times L$, onde $L = 2^\gamma L_{\min}$. Neste caso, o número de parâmetros da *waveletconcat* é $K = 2^\gamma L_{\min} Q$ e o número de quadros é dado por

$$N = 1 + (T - L_{\min}) / S_{\min} \quad (3.3)$$

onde $T_{\mathbf{a}}$ é o número de elementos em \mathbf{a} . Este *front end* é chamado na literatura de *waveletconcat* (MORAIS et al., 2010).

Outra alternativa para organizar os coeficientes *wavelet* é calculando a energia média de cada coeficiente. Este *front end* é chamado de *waveletenergy*, e semelhante ao *waveletconcat*, este tem que tratar com sinais de diferentes frequências de amostragem. A principal diferença entre esses *front ends* é que, em vez de concatenar todos os coeficientes, *waveletenergy* representa \mathbf{X} por meio da energia E em cada banda de frequência obtida pela decomposição *wavelet*. A energia é estimada no estilo “média móvel” (*moving average*), ou seja, calcula-se a energia para curtos intervalos ou quadros (mas deve-se evitar confusão com o já utilizado termo “quadro”). Por exemplo, assumindo $L_{\min} = S_{\min} = 1$ e novamente uma matriz \mathbf{X} de dimensão 6×1000 , temos uma matriz 6×4 , onde a primeira linha contém a energia média do coeficiente *wavelet* \mathbf{a} calculado a partir da tensão da fase A, logo para o caso da *waveletenergy* $K = 24$ e $N = 125$.

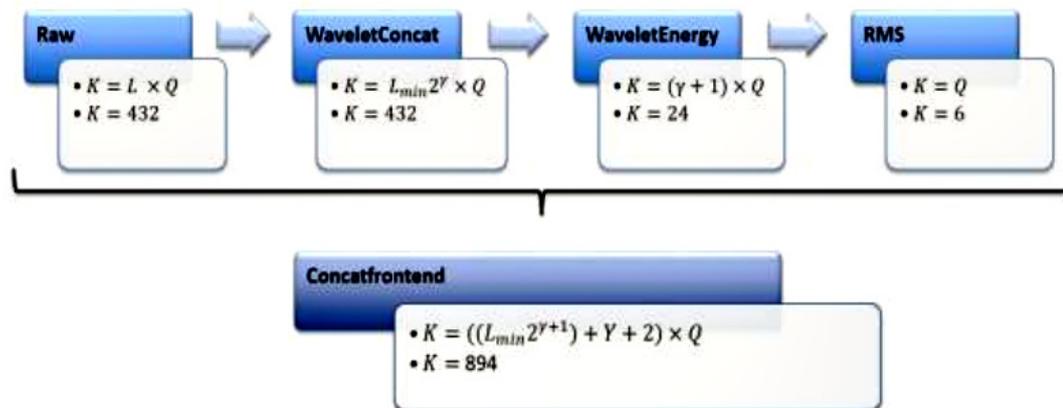
3.2.4 Concatenação de *fronts ends*

Nas subseções anteriores foram citados os *fronts ends* utilizados para comparação neste trabalho, assim como todas suas características e parâmetros que incidem diretamente no resultados de desempenho da classificação de faltas em linhas de transmissão, com o objetivo de alcançar a melhor acuraria (HOMCI et al., 2016) sugere a concatenação dos *fronts ends raw*, *waveletconcat*, *waveletenergy* e RMS, para gerar um único conjunto de sequências.

Denominado de *concatfrontend*, o processo é realizado em duas etapas, a primeira é realizado o processo de concatenação por sinal, onde os *front ends* são calculados para cada sinal (tensão e corrente; fases A, B e C) e agrupados na ordem definida previamente. Para tal, o tamanho do quadro L e o deslocamento do quadro S devem ser calculados, onde $L = L_{\min} 2^\gamma$ e $S = S_{\min} 2^\gamma$. Com os parâmetros de janelamento e decomposição *wavelet* definidos, é possível gerar todos os *front ends* para cada sinal e concatená-los numa matriz na seguinte ordem: primeiramente as tensões das fases A, B e C; em seguida as correntes das fases A, B e C.

A figura 18 ilustra a concatenação dos *front ends*, utilizando os seguintes parâmetros, tamanho da janela $L_{\min} = 9$ e o deslocamento do quadro $S_{\min} = 4$, bem como o número de elementos $Q = 6$ e o nível de decomposição *wavelet* = 3, observa-se que dependendo do *frontend*, o número de parâmetros K cresce muito rápido, gerando matrizes de alta dimensionalidade.

Figura 18 – Processo de concatenação dos *front ends*, mostrando o número de parâmetros K para cada *front end*, considerando $L_{min} = 9$ e $S_{min} = 4$.



Fonte: Próprio autor

O *concatfrontend* gera uma matriz com um grande número de parâmetros, ocasionando um elevado custo computacional quando submetido a um classificador. Como consequência desse processo, temos o surgimento de um problema que várias aplicações possuem, que é geração de uma sequência com alto grau de dimensionalidade (PRIDDY; KELLER, 2005). Por esse motivo, (HOMCI et al., 2016) adota técnicas de seleção de parâmetros para reduzir a dimensão dos dados de entrada selecionando os parâmetros mais relevantes, antes do estágio de reconhecimento de padrões.

4 AVALIAÇÃO EXPERIMENTAL DA PROPOSTA

Este capítulo tem como objetivo descrever os experimentos realizados e os resultados obtidos para classificação de eventos em sistemas de potência usando a arquitetura FBSC, o classificador KNN-DTW e a base de dados UFPAFaults.

4.1 Configuração dos experimentos

É assumido ao longo das simulações o uso de um gatilho (*trigger*) (MCEACHERN, 1990; ENGLERT; STENZEL, 2002) que descarta as amostras quando a forma de onda não apresenta qualquer tipo de anomalia, e passa aos estágios subsequentes somente o segmento onde a falta foi detectada. Mais especificamente, considera-se um algoritmo de gatilho perfeito, com os pontos de início e fim da falta sendo diretamente obtidos a partir de cada sequência armazenada na base de dados UFPAFaults.

Foram utilizados os *front ends* waveletconcat, waveletenergy, raw e RMS. A wavelet mãe utilizada foi Daubechies 4 com $\gamma = 3$ níveis de decomposição. Os valores de L_{\min} e S_{\min} foram fixados em 9 e 4, respectivamente de acordo a parametrização adotada em (MORAIS et al., 2010).

Os classificadores convencionais utilizados para FBSC, são todos pertencentes ao pacote WEKA (WITTEN; FRANK, 2005) descritos no Capítulo 2. Os valores que definem os modelos dos classificadores foram escolhidos através de um procedimento de seleção automática de modelo (*automatic model selection*) (CASTLE J., 2011).

Após vários testes, a arquitetura mais robusta para rede neural foi com apenas uma camada escondida com 160 neurônios fixados em H, taxa de aprendizagem L de 0.9 e taxa de momentum M de 0.4, sendo que o número de épocas fixado em 1500.

Para o classificador KNN, foi utilizado a classe IBK do WEKA e o parâmetro K (número de vizinhos mais próximos) foi variado em 1, 3, 5, ..., 15, sendo eleito como melhor valor K igual a 1.

Para o classificador SVM foi utilizada a seguinte biblioteca LibSVM (EL-MANZALAWY; HONAVAR, 2005). A função kernel utilizada foi a polinomial de primeira ordem. A escolha desse kernel é baseada em trabalhos anteriores. Os parâmetros otimizados foram gama G (largura da função kernel) e penalidade de erro C, sendo que os melhores valores foram 0.01 e 100, respectivamente.

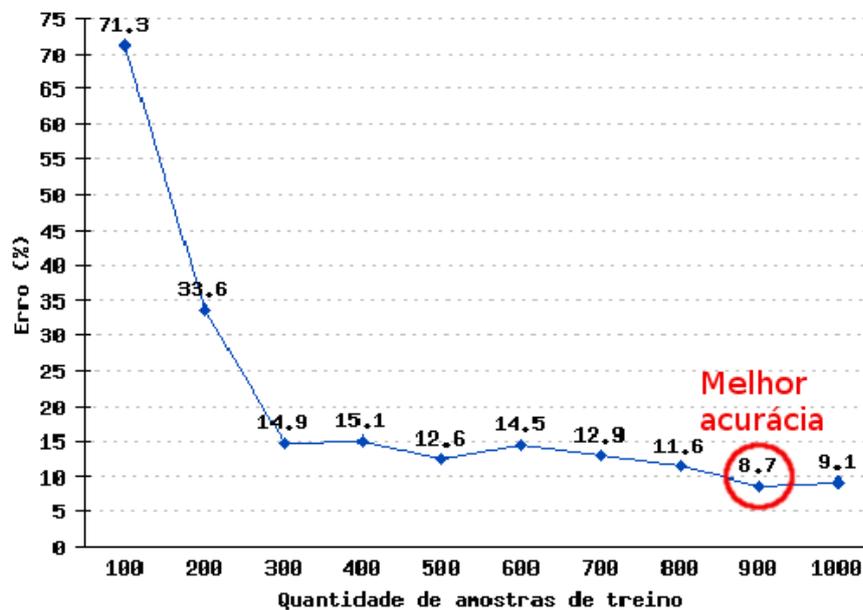
Em relação à árvore de decisão foi utilizada a classe do WEKA, denominada de J4.8. Os parâmetros C (fator de confiança para poda) e M (o número mínimo de exemplos por folha) desta classe foram variados, sendo os melhores valores foram 0.74 e 10, respectivamente. Note que as mesmas letras identificam parâmetros distintos, dependendo do classificador.

4.2 Resultados para classificação de sequências utilizando KNN-DTW

O algoritmo KNN-DTW foi implementado na linguagem de programação Java ¹. O primeiro experimento realizado para este classificador foi para verificar a quantidade de amostras necessárias para o treinamento do mesmo. Este estudo é denominado de curva de complexidade amostral.

A figura 19 exibe a curva de complexidade amostral para o classificador KNN-DTW. Observamos que a classificação com 900 amostras de treino, é suficiente para atingir a melhor acurácia entre a fase de treino, que utilizou as amostras variando de 100 a 1000 faltas devidamente rotuladas. Contudo, os resultados do classificador KNN-DTW apresentados adiante consideram que o mesmo foi treinado e testado com 1000 amostras, e comparado com a classificação quadro a quadro (FBSC), replicado com 1000 amostras para treino e testes. Ressaltando que o valor de K (quantidade de vizinhos) do KNN-DTW variou entre 1 e 15, sendo o melhor resultado com apenas 1 vizinho.

Figura 19 – Curva de complexidade amostral para o classificador KNN-DTW, considerando 1 vizinho mais próximo.



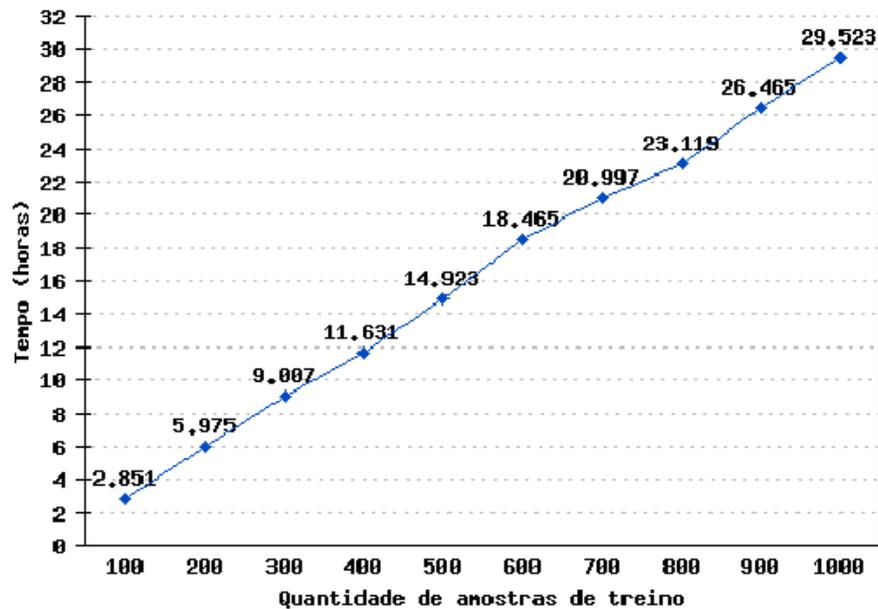
Fonte: Gráfico gerado pelo autor

O custo computacional utilizando KNN-DTW, exibido na figura 20, mostrou um crescimento no eixo do tempo proporcional ao aumento das amostras 100, 200, 300, ..., 1000, respectivamente. As amostras iniciais de treino contendo 100 faltas do tipo curto-circuito, revelou um tempo de classificação acima 2 horas, esboçando um comportamento próximo a um crescimento linear, atingindo valores inferior a 30 horas para classificar 1000 amostras de teste, representando

¹ O algoritmo KNN-DTW implementado assim como a base de dados UFPAFaults estão disponíveis em: <https://github.com/Bruno1307/knn-dtw/>

faltas do tipo curto-circuito em linhas de transmissão. O tempo foi medido usando o comando *time* do sistema operacional Linux, todos os experimentos foram executados em um computador pessoal Intel core I7 1.8 GHz e 16GB de memória RAM.

Figura 20 – Custo computacional utilizando DTW

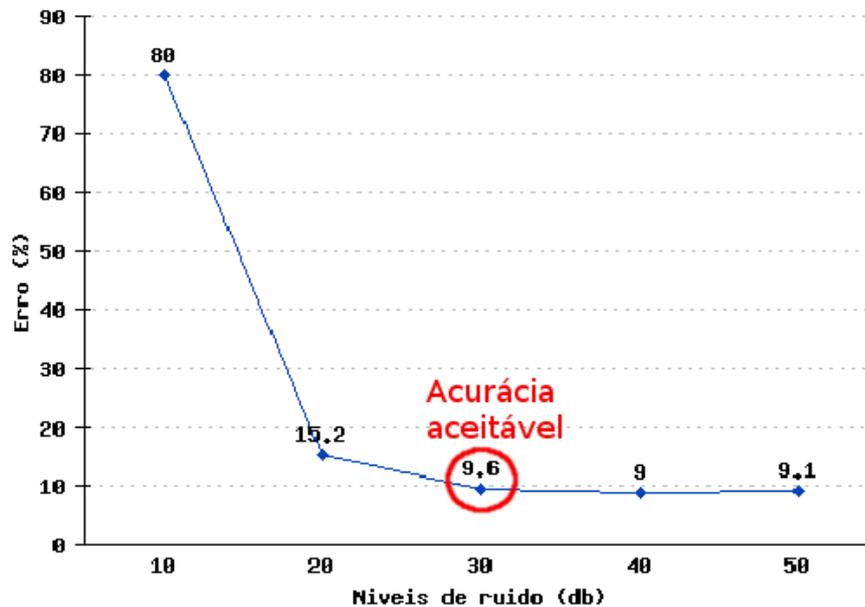


Fonte: Gráfico gerado pelo autor

Para simular as diversas interferências indesejáveis presentes nas linhas de transmissão, adicionou-se ruído branco gaussiano na base de teste. Este sinal é um processo aleatório estacionário ergódico com média igual a zero e densidade espectral de potência constante para todas as frequências, por essa razão denominado “ruído branco”, análogo a luz branca que tem todos os comprimentos de ondas visíveis (MARMARELIS, 2004). Uma amostra do processo gaussiano possui uma distribuição de probabilidade gaussiana para sua amplitude.

A figura 21 esboça o impacto na robustez do classificador KNN-DTW após a adição de ruído branco gaussiano nas formas de onda de tensão e corrente. O classificador foi treinado com formas de ondas sem qualquer tipo de ruído e em seguida testado sob uma condição de razão sinal ruído (SNR - *Signal-to-Noise Ratio*) variando em 10, 20, 30, 40 e 50 dB.

Figura 21 – Avaliação da robustez do classificador KNN-DTW em termos de taxa de erro sob uma condição de razão sinal ruído (SNR - *Signal-to-Noise Ratio*) variando em 10, 20, 30, 40 e 50 dB. O conjunto de treino foi composto por 1000 amostras sem ruído.



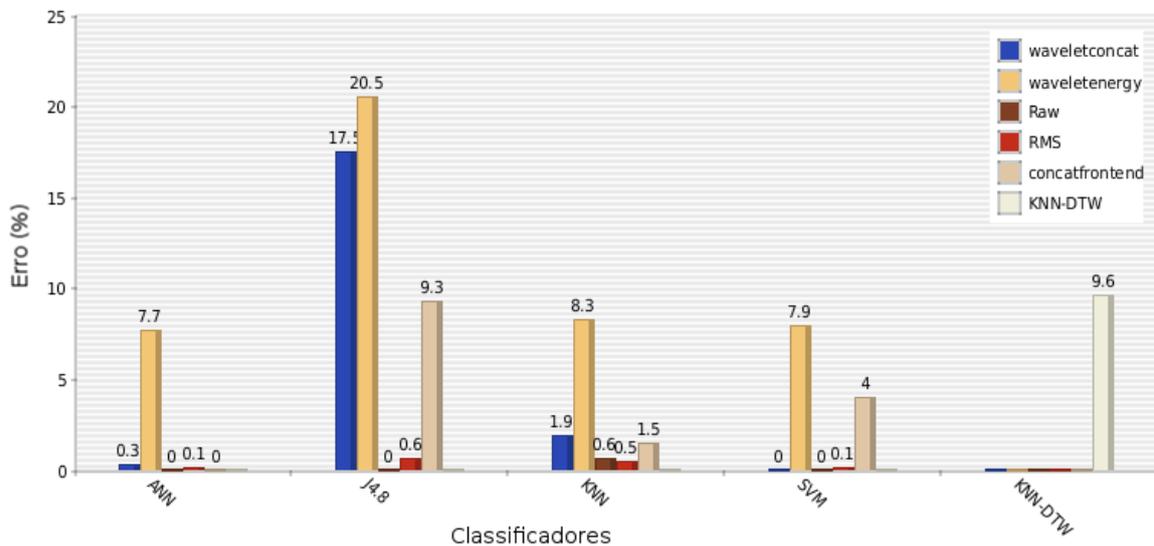
Fonte: Gráfico gerado pelo autor

Analisando a figura 21 pode-se inferir que a taxa de erro do classificador KNN-DTW mantém-se estável a partir de 30 db, o que mostra uma capacidade de generalização adequada mesmo na presença de ruídos deste classificador, ou seja o algoritmo KNN-DTW classificou 1000 amostras representando faltas com a diferentes níveis de ruído, onde 10 db foi máximo de ruído em 1000 amostras de teste e 50 db a menor quantidade de ruído, atingindo acurácia próxima a classificação de 1000 amostras sem ruído, porém com um nível intermediário de 30 db (ruído) a curva se estabiliza, exibindo uma acurácia em torno do valores de classificação sem ruído, logo adota-se uma "acurácia aceitável" a partir de 30 db de ruído.

4.3 Comparando a Arquitetura FBSC com o classificador KNN-DTW

A figura 22 apresenta uma comparação das taxas de erros da arquitetura FBSC, considerando os classificadores convencionais ANN, J4.8, KNN e SVM e os front ends raw, RMS, waveletconcat, waveletenergy e concatfrontend, e do classificador KNN-DTW. Observa-se que o classificador KNN-DTW apresenta uma taxa de erro mais alta que a maioria dos classificadores da arquitetura FBSC. Contudo, vale ressaltar que diferente da arquitetura FBSC o algoritmo KNN-DTW é isento de procedimentos que segmentação, *front end*, classificador convencional e sistema de *Voting*, principalmente se consideramos todo o seu *pipeline* (extração e seleção de parâmetros). Enquanto que o algoritmo KNN-DTW executa a fase de classificação dos dados.

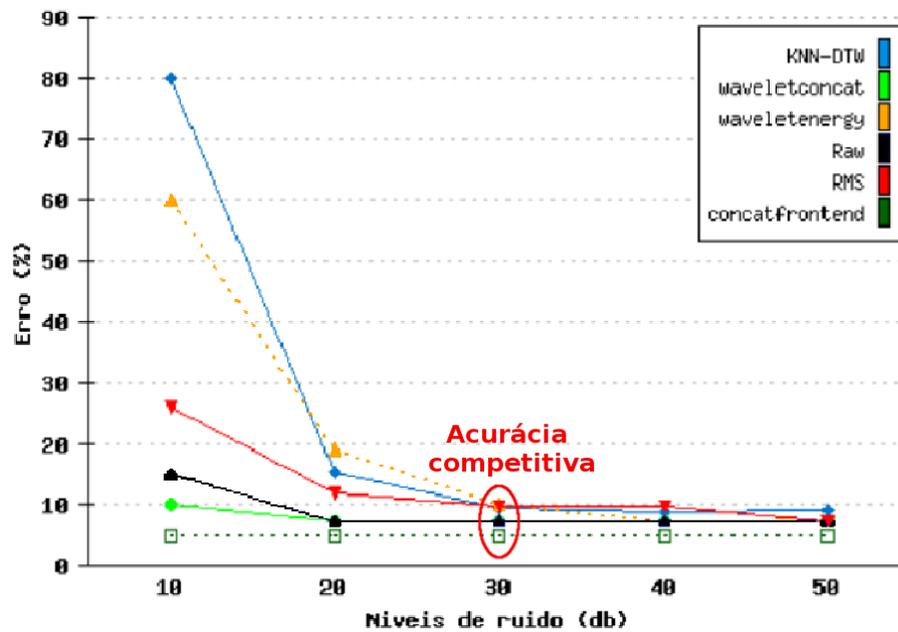
Figura 22 – Comparação das taxas de erro dos classificadores convencionais da arquitetura FBSC com o classificador KNN-DTW. Os valores de $L = L_{\min}$ e $S = S_{\min}$ dos fronts ends testados foram 9 e 4, respectivamente



Fonte: Gráfico gerado pelo autor

A figura 23 apresenta uma comparação do classificador KNN convencional da arquitetura FBSC, isto é, o que utiliza a distância Euclidiana como medida de similaridade, com o classificador KNN-DTW em termos de suas acurácias (taxa de erro) em um cenário com ruído. Para o classificador KNN convencional foram utilizados os *front ends* raw, RMS, waveletconcat, waveletenergy e concatfrontend. Os valores da razão sinal ruído variaram entre 10 e 50 dB.

Figura 23 – Comparação, em termos de acurácia (taxa de erro), da arquitetura FBSC e do classificador KNN-DTW em um cenário com ruído. O classificador da arquitetura FBSC utilizado foi o KNN convencional que utiliza a distância euclidiana como medida de similaridade. Os front ends usados foram waveletconcat, waveletenergy, Raw, RMS e concatfrontend, considerando $L = L_{min} = 9$ e $S = S_{min} = 4$.



Fonte: Gráfico gerado pelo autor

Objetivo da figura 23 é analisar o comportamento da arquitetura FBSC e do classificador KNN-DTW, em uma situação onde os dados de treino não apresentam ruído e os de teste apresentam ruído branco Gaussiano simulando situações reais dos Sistemas de Potência. O classificador KNN convencional da arquitetura FBSC, foi escolhido para ser comparado com o KNN-DTW devido sua proximidade com este último, já que a diferença entre esses classificadores é exatamente na medida de similaridade adotada. Observar-se que o classificador KNN-DTW, apesar de apresentar uma taxa de erro (80%) elevada para um cenário com grau de ruído bastante alto (10 dB), este consegue a partir de 30 dB de ruído ser competitivo em termos de acurácia em relação ao classificador KNN convencional da arquitetura FBSC.

Os resultados apresentados nas figuras 22 e 23 podem ser considerados como um importante indicativo da viabilidade do uso do KNN-DTW para classificação de faltas, visto que, diferente dos classificadores da arquitetura FBSC, este não possui muitos graus de liberdade na concepção do seu modelo. Em outras palavras, mesmo considerando que alguns classificadores da arquitetura FBSC tenham apresentado taxas de erro de 0% (vide figura 22), o classificador KNN-DTW é uma alternativa viável na tarefa de classificação de faltas por dois motivos: i) é um classificador que apresenta robustez ao ser testado em um cenário com ruído; ii) é um classificador que exige pouca parametrização na concepção do seu modelo.

5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Ao longo deste trabalho investigou-se uma importante classe de eventos que compromete a QEE fornecida pelos SEP: as faltas do tipo curto-circuito em linhas de transmissão. A classificação dessas faltas em um cenário *off-line*, isto é, em um cenário pós-falta, é caracterizada como um problema de classificação de sequências, uma tarefa de classificação especial onde os dados de entrada são representados por uma matriz que possui tamanho variável. Nesse contexto, duas abordagens para classificação de faltas foram apresentadas: a arquitetura de classificação de sequências baseada em quadros (FBSC), já utilizada na literatura; e a aplicação do algoritmo K-vizinho mais próximo tendo como medida de similaridade o alinhamento temporal dinâmico (DTW).

A arquitetura FBSC, replicada neste trabalho utiliza algoritmos convencionais, ou seja, que lidam com entradas de tamanho fixo tais como redes neurais e máquinas de vetores de suporte na tarefa de classificação de faltas. Para isso, adota uma etapa de pré-processamento ou *front end* que converter ou organizar as amostras das sequências de tamanho variável em segmentos (quadros) de tamanho fixo, podem ser processados pelos classificadores convencionais. Acontece que esta arquitetura de classificação, possui elevado grau de parametrização, tanto dos *front ends*, quanto dos algoritmos de aprendizado de máquina, tornando o processo de escolha do melhor modelo um procedimento custoso e não trivial.

Nesse contexto, como alternativa à arquitetura FBSC, foi utilizado na classificação de faltas o algoritmo KNN-DTW, que diferente do primeiro, consegue lidar com amostras de tamanho diferentes, tendo como principal característica baixa parametrização para encontrar seu melhor modelo.

Nos experimentos realizados, utilizou-se a base de dados UFPAFaults composta por simulações, baseada em um modelo de sistema elétrico de potência real, de faltas do tipo curto-circuito em linhas de transmissão. Os classificadores convencionais adotados na arquitetura FBSC são todos pertencentes ao pacote de mineração de dados WEKA e os *front ends* testados são os mais utilizados na literatura.

Todos os resultados comparados, foram referentes à acurácia (taxa de erro) das técnicas apuradas, indicam que em um cenário com ruído e com a necessidade de um classificador com baixa parametrização, o algoritmo KNN-DTW mostra-se como uma alternativa viável, mesmo que sua acurácia tenha sido inferior se comparado à algumas técnicas da arquitetura FBSC.

Vale ressaltar, que é necessária uma investigação mais aprofundada que permita avaliar e comparar em termos de custo computacional a arquitetura FBSC e o algoritmo KNN-DTW. Entretanto, os resultados apresentados e discutidos reforçam a principal contribuição desta dissertação de mestrado que é de apresentar o KNN-DTW como uma alternativa à arquitetura FBSC na tarefa de classificação de faltas. Outros modelos, com o mesmo propósito de lidar diretamente com as sequências de tamanho variável, devem ser avaliados como é o caso das

Cadeias Ocultas de Markov (*Hidden Markov Models*-HMM).

5.1 Trabalhos Futuros

Como sugestões para trabalhos futuros relacionados com as contribuições apresentadas nesta dissertação, é possível citar:

- A base de dados UFPAFaults se restringe apenas a faltas do tipo curto-circuito em linhas de transmissão. Neste caso tal base de dados poderia ser expandida para outros eventos de QEE como descargas atmosféricas e chaveamento de banco de capacitores.
- Avaliar outras técnicas capazes de lidar diretamente com sequências de tamanho diferentes tais como as cadeia ocultas de Markov (ESMAEL et al., 2012).
- Avaliar outras implementações do classificador KNN-DTW que permita aumentar sua acurácia como a que é apresentada em (PETITJEAN et al., 2016).
- Comparar, em termos de custo computacional, a arquitetura de classificação de faltas baseada em quadros com o classificador KNN-DTW.

REFERÊNCIAS

- ABDOLLAHI, A.; SEYEDTABAI, S. Comparison of fourier amp; wavelet transform methods for transmission line fault classification. In: **2010 4th International Power Engineering and Optimization Conference (PEOCO)**. [S.l.: s.n.], 2010. p. 579–584.
- AGUILERA, C.; ORDUNA, E.; RATA, G. Fault detection, classification and faulted phase selection approach based on high-frequency voltage signals applied to a series-compensated line. **IEE Proceedings - Generation, Transmission and Distribution**, v. 153, n. 4, p. 469–475, July 2006. ISSN 1350-2360.
- BERMEJO, P. e. a. Fast wrapper feature subset selection in high-dimensional datasets by means of filter re-ranking. **Knowledge-Based Systems**, p. 35–44, 2012.
- BERMEJO P.; GAMEZ, J. A. P. J. M. Improving incremental wrapper-based subset selection via replacement and early stopping. **International Journal of Pattern Recognition and Artificial Intelligence**, p. 20, 2011.
- CAMPOS, G. O. et al. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. **Data Mining and Knowledge Discovery**, v. 30, n. 4, p. 891–927, Jul 2016. ISSN 1573-756X. Disponível em: <<https://doi.org/10.1007/s10618-015-0444-8>>.
- CARDOSO, C. et al. Hierarchical agglomerative clustering of short-circuit faults in transmission lines. In: **2008 10th Brazilian Symposium on Neural Networks**. [S.l.: s.n.], 2008. p. 87–92. ISSN 1522-4899.
- CASTLE J., D. J. . H. D. Evaluating automatic model selection. **Journal of Time Series Econometrics**, v. 3(1), 2011.
- CHANDRASHEKAR G.; SAHIN, F. A survey on feature selection methods. **Computers and Electrical Engineering**, n. 1, p. 16–28, 2014.
- CHATTOPADHYAY, S.; MITRA, M.; SENGUPTA, S. Electric power quality. In: _____. **Electric Power Quality**. Dordrecht: Springer Netherlands, 2011. p. 5–12. ISBN 978-94-007-0635-4. Disponível em: <https://doi.org/10.1007/978-94-007-0635-4_2>.
- CHUNG, J. e. a. Power disturbance classifier using a rule-based method and wavelet packet-based hidden markov model. **IEEE Transactions on Power Delivery**, v. 17, n. 1, p. 233–241, 2002.
- DASGUPTA A.; DEBNATH, S. D. A. Transmission line fault detection and classification using cross-correlation and k-nearest neighbor. **International Journal of Knowledge-based and Intelligent Engineering Systems**, v. 19, n. 1, p. 183–189, 2015.
- EL-MANZALAWY, Y.; HONAVAR, V. Wlsvm: Integrating libsvm into weka environment. **Software available at <http://www.cs.iastate.edu/yasser/wlsvm>**, 2005.
- ENGLERT, H.; STENZEL, J. Automated classification of power quality events using speech recognition techniques. In **14th Power Systems Computation Conference**, 2002.
- ESMAEL, B. et al. Improving time series classification using hidden markov models. In: **2012 12th International Conference on Hybrid Intelligent Systems (HIS)**. [S.l.: s.n.], 2012. p. 502–507.

- FACELI, K. e. a. Inteligência artificial: Uma abordagem de aprendizado de máquina. **Rio de Janeiro: LTC**, 2011.
- FATHABADI, H. Novel filter based ann approach for short-circuit faults detection, classification and location in power transmission lines. **International Journal of Electrical Power Energy Systems**, v. 74, n. Supplement C, p. 374 – 383, 2016. ISSN 0142-0615. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S014206151500335X>>.
- FATHABADI, H. Novel filter based ann approach for short-circuit faults detection, classification and location in power transmission lines. **International Journal of Electrical Power and Energy Systems**, v. 74, p. 374–383, 2016.
- FONTES, C. H.; PEREIRA, O. Pattern recognition in multivariate time series – a case study applied to fault detection in a gas turbine. **Engineering Applications of Artificial Intelligence**, v. 49, n. Supplement C, p. 10 – 18, 2016. ISSN 0952-1976. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0952197615002602>>.
- GELER, Z. et al. Impact of the sakoe-chiba band on the dtw time series distance measure for knn classification. In: _____. **Knowledge Science, Engineering and Management: 7th International Conference, KSEM 2014, Sibiu, Romania, October 16-18, 2014. Proceedings**. Cham: Springer International Publishing, 2014. p. 105–114. ISBN 978-3-319-12096-6. Disponível em: <https://doi.org/10.1007/978-3-319-12096-6_10>.
- GIORGINO, T. Computing and visualizing dynamic time warping alignments in r: The dtw package. **Journal of Statistical Software, Articles**, v. 31, n. 7, p. 1–24, 2009. ISSN 1548-7660. Disponível em: <<https://www.jstatsoft.org/v031/i07>>.
- GOWRISHANKAR M.; NAGAVENI, P. B. P. Transmission line fault detection and classification using discrete wavelet transform and artificial neural network. **Middle-East Journal of Scientific Research**, v. 24, n. 4, p. 1112–1121, 2016.
- HALL EIBE FRANK, G. H. B. P. P. R. I. H. W. M. The weka data mining software: an update. **International Journal on Electrical Engineering and Informatics**, ACM SIGKDD explorations newsletter, v. 11, n. 1, p. 10–18, 2009.
- HAN J.; KAMBER, M. P. J. Data mining: Concepts and techniques. [s.l.]. **Middle-East Journal of Scientific Research**, 2012.
- HAYKIN, S. Redes neurais: Principios e pratica. **2. Ed. Porto Alegre: Bookman**, 2001.
- HOMCI, M. et al. A new strategy based on feature selection for fault classification in transmission lines. In: _____. **Advances in Artificial Intelligence - IBERAMIA 2016: 15th Ibero-American Conference on AI, San José, Costa Rica, November 23-25, 2016, Proceedings**. Cham: Springer International Publishing, 2016. p. 376–387. ISBN 978-3-319-47955-2. Disponível em: <https://doi.org/10.1007/978-3-319-47955-2_31>.
- HOSSEINI, K. Short circuit fault classification and location in transmission lines using a combination of wavelet transform and support vector machines. **International Journal on Electrical Engineering and Informatics**, v. 7, n. 2, p. 353–365, 2015.
- KEOGH, E.; RATANAMAHATANA, C. A. Exact indexing of dynamic time warping. **Knowledge and Information Systems**, v. 7, n. 3, p. 358–386, Mar 2005. ISSN 0219-3116. Disponível em: <<https://doi.org/10.1007/s10115-004-0154-9>>.

LIVANI H.; EVRENOSOGLU, C. Y. A machine learning and wavelet-based fault location method for hybrid transmission lines. **Smart Grid, IEEE Transactions on**, v. 5, n. 1, p. 51–59, 2014.

MARMARELIS, V. Z. Appendix ii: Gaussian white noise. In: _____. **Nonlinear Dynamic Modeling of Physiological Systems**. John Wiley Sons, Inc., 2004. p. 499–501. ISBN 9780471679370. Disponível em: <<http://dx.doi.org/10.1002/9780471679370.app2>>.

MCEACHERN, A. A floating-window algorithm for detecting certain power line faults that disrupt sensitive electronic loads. **IEEE Transactions on Instrumentation and Measurement**, v. 39, p. 112–115, 1990.

MORAIS, J. et al. Data mining applied to the electric power industry: Classification of short-circuit faults in transmission lines. In: **Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007)**. [S.l.: s.n.], 2007. p. 943–948. ISSN 2164-7143.

MORAIS, J. et al. A framework for evaluating automatic classification of underlying causes of disturbances and its application to short-circuit faults. **IEEE Transactions on Power Delivery**, v. 25, n. 4, p. 2083–2094, Oct 2010. ISSN 0885-8977.

OSHIRO T. M.; PEREZ, P. S. B. J. A. How many trees in a random forest? in: **Machine Learning and Data Mining in Pattern Recognition**, p. 154–168, 2012. Berlin: Springer Berlin Heidelberg.

PATEL, A. C. V. Wavelet transform application to fault classification on transmission line. **INDIAN JOURNAL OF APPLIED RESEARCH**, v. 5, 2015. ISSN 2249-555X. Issue: 2.

PEREIRA, S.; MORETO, M. A wavelet based tool to assist the automated analysis of waveform disturbances records in power generators. **IEEE Latin America Transactions**, v. 14, n. 8, p. 3621–3629, Aug 2016. ISSN 1548-0992.

PETITJEAN, F. et al. Dynamic time warping averaging of time series allows faster and more accurate classification. In: **2014 IEEE International Conference on Data Mining**. [S.l.: s.n.], 2014. p. 470–479. ISSN 1550-4786.

PETITJEAN, F. et al. Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. **Knowledge and Information Systems**, v. 47, n. 1, p. 1–26, Apr 2016. ISSN 0219-3116. Disponível em: <<https://doi.org/10.1007/s10115-015-0878-8>>.

PRIDDY, K. L.; KELLER, P. E. The curse of dimensionality. **Washington: SPIE - The International Society for Optical Engineering**, Artificial Neural Networks: An Introduction, TT68, p. 180, Aug 2005.

QUINLAN, J. C4.5: Programs for machine learnin. **Morgan Kaufmann**, 1993.

REDDY, M. J. B.; RAJESH, D. V.; MOHANTA, D. Robust transmission line fault classification using wavelet multi-resolution analysis. **Computers Electrical Engineering**, v. 39, n. 4, p. 1219 – 1247, 2013. ISSN 0045-7906. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0045790613000578>>.

SAMANTARAY, S. A systematic fuzzy rule based approach for fault classification in transmission lines. **Applied Soft Computing**, v. 13, n. 2, p. 928 – 938, 2013. ISSN 1568-4946. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1568494612004309>>.

SAWATPIPAT, P.; TAYJASANANT, T. Fault classification for thailand's transmission lines based on discrete wavelet transform. In: **ECTI-CON2010: The 2010 ECTI International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology**. [S.l.: s.n.], 2010. p. 636–640.

SCOTT, W.; LIU, T. Alternative transients program atp rule book. **Portland: Canadian/American EMTP User Group**, 1992.

SHAIK, A. G.; PULIPAKA, R. R. V. A new wavelet based fault detection, classification and location in transmission lines. **International Journal of Electrical Power Energy Systems**, v. 64, n. Supplement C, p. 35 – 40, 2015. ISSN 0142-0615. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0142061514004232>>.

SILVA, D. F.; BATISTA, G. E. A. P. A. Speeding up all-pairwise dynamic time warping matrix calculation. In: _____. **Proceedings of the 2016 SIAM International Conference on Data Mining**. [s.n.]. p. 837–845. Disponível em: <<http://epubs.siam.org/doi/abs/10.1137/1.9781611974348.94>>.

SILVA, K. M. et al. Haar wavelet-based method for fast fault classification in transmission lines. In: **2006 IEEE/PES Transmission Distribution Conference and Exposition: Latin America**. [S.l.: s.n.], 2006. p. 1–5.

SILVA, K. M.; SOUZA, B. A.; BRITO, N. S. D. Fault detection and classification in transmission lines based on wavelet transform and ann. **IEEE Transactions on Power Delivery**, v. 21, n. 4, p. 2058–2063, Oct 2006. ISSN 0885-8977.

SINGH, M.; PANIGRAHI, B. K.; MAHESHWARI, R. P. Transmission line fault detection and classification. In: **2011 International Conference on Emerging Trends in Electrical and Computer Technology**. [S.l.: s.n.], 2011. p. 15–22.

SUDHA, G. A comparison between different approaches for fault classification in transmission lines. **IET Conference Proceedings**, Institution of Engineering and Technology, p. 398–403(5), January 2007. Disponível em: <http://digital-library.theiet.org/content/conferences/10.1049/ic_20070645>.

TANG, B.; HE, H. Enn: Extended nearest neighbor method for pattern recognition [research frontier]. **IEEE Computational Intelligence Magazine**, v. 10, n. 3, p. 52–60, Aug 2015. ISSN 1556-603X.

TANG, H. et al. End-to-end neural segmental models for speech recognition. PP, 08 2017.

TAYEB, E. B. M.; RHIM, O. A. A. A. Transmission line faults detection, classification and location using artificial neural network. In: **2011 International Conference Utility Exhibition on Power and Energy Systems: Issues and Prospects for Asia (ICUE)**. [S.l.: s.n.], 2011. p. 1–5.

VALENTI, A.; GIUFFRIDA, S.; LINGUANTI, F. Decision trees analysis in a low tension real estate market: The case of troina (italy). In: _____. **Computational Science and Its Applications – ICCSA 2015: 15th International Conference, Banff, AB, Canada, June**

22-25, 2015, Proceedings, Part III. Cham: Springer International Publishing, 2015. p. 237–252. ISBN 978-3-319-21470-2. Disponível em: <https://doi.org/10.1007/978-3-319-21470-2_17>.

VETTERLI, I.; KOVAČEVIĆ, J. Wavelets and subband coding. **Prentice Hal**, 1995.

WITTEN, I.; FRANK, E. Data mining: practical machine learning tools and techniques with java implementations. **Morgan Kaufmann**, 2005.

XI, X. et al. Fast time series classification using numerosity reduction. In: **Proceedings of the 23rd International Conference on Machine Learning.** New York, NY, USA: ACM, 2006. (ICML '06), p. 1033–1040. ISBN 1-59593-383-2. Disponível em: <<http://doi.acm.org/10.1145/1143844.1143974>>.

YADAV, A.; DASH, Y. An overview of transmission line protection by artificial neural network: Fault detection, fault classification, fault location, and fault direction discrimination. **Adv. Artif. Neu. Sys.**, Hindawi Publishing Corp., New York, NY, United States, v. 2014, p. 12:12–12:12, jan. 2015. ISSN 1687-7594. Disponível em: <<http://dx.doi.org/10.1155/2014/230382>>.

YADAV, A.; SWETAPADMA, A. Combined dwt and naive bayes based fault classifier for protection of double circuit transmission line. In: **International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014).** [S.l.: s.n.], 2014. p. 1–6.

YOMARA, P. et al. Amazontp: Uma ferramenta para simulação de eventos de qualidade de energia. In: **Proceedings of the XXVIII Iberian Latin American Congresso on Computational Methods in Engineering.** [S.l.: s.n.], 2006.

YONG D.; BHOWMIK, S. M. F. D. An effective power quality classifier using wavelet transform and support vector machines. **Expert Systems with Applications**, v. 42, n. 15–16, p. 6075–6081, 2015.

YURTMAN, A.; BARSHAN, B. Detection and evaluation of physical therapy exercises by dynamic time warping using wearable motion sensor units. In: _____. **Information Sciences and Systems 2013: Proceedings of the 28th International Symposium on Computer and Information Sciences.** Cham: Springer International Publishing, 2013. p. 305–314. ISBN 978-3-319-01604-7. Disponível em: <https://doi.org/10.1007/978-3-319-01604-7_30>.

ZHANG, N.; KEZUNOVIC, M. A real time fault analysis tool for monitoring operation of transmission line protective relay. **Electric Power Systems Research**, v. 77, n. 3, p. 361 – 370, 2007. ISSN 0378-7796. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0378779606000800>>.

ZHONGMING, Y.; BIN, W. A review on induction motor online fault diagnosis. In: **Proceedings IPEMC 2000. Third International Power Electronics and Motion Control Conference (IEEE Cat. No.00EX435).** [S.l.: s.n.], 2000. v. 3, p. 1353–1358 vol.3.

ZHOU, Q. et al. Structure damage detection based on random forest recursive feature elimination. **Mechanical Systems and Signal Processing**, v. 46, n. 1, p. 82 – 90, 2014. ISSN 0888-3270. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0888327013006821>>.

ANEXO A – CÓDIGO DO KNN-DTW EM JAVA

Trecho do código que define a classe utilizada para implementar o algoritmo KNN.

```

public class KNN extends Classifier {

    protected List<Record> trainDS;
    public int cont = 0;

    @Override
    public int classify(Record record) {
        int ret = 0;
        List< Pair<Integer, Double>> candidates = new
LinkedList<Pair<Integer, Double>>();
        for (Record trained : trainDS) {
            double similarity = metric.compute(trained, record);
            candidates.add(new Pair<Integer, Double>(trained.label, similarity));
        }
        Collections.sort(candidates);
        int stats[] = new int[classes + 1];
        int arrayLength = kValue < candidates.size() ? kValue :
candidates.size();
        int menor = 0;
        Double menorDistancia = 100000.0;
        for (int i = 0; i < arrayLength; i++) {
            int thisLabel = candidates.get(i).key;
            if (candidates.get(i).value < menorDistancia) {
                menorDistancia = candidates.get(i).value;
                menor = candidates.get(i).key;
            }
            stats[thisLabel]++;
        }
        int max = 0;
        ret = menor;
        cont++;
        System.out.println("Classificacao " + cont + ": " + ret);
        return ret;
    }
}

```

Classe Java que implementa o cálculo do alinhamento temporal dinâmico (DTW).

```

public class DTW implements Metric {

    protected Double[] seq1;
    protected Double[] seq2;
    protected int[][] warpingPath;
    protected int n;
    protected int m;
    protected int K;
    protected double warpingDistance;

    public DTW(float[] sample, float[] templete) {
        K = 1;
        warpingPath = new int[n + m][2];
        warpingDistance = 0.0;
    }

    public DTW() {
        K = 1;
        warpingPath = new int[n + m][2];
        warpingDistance = 0.0;
    }

    @Override
    public double compute(Record r1, Record r2) {
        String[] treinoTamanho = r1.toString().split(",");
        String[] testeTamanho = r2.toString().split(",");
        seq1 = new Double[treinoTamanho.length + 1];
        seq2 = new Double[testeTamanho.length + 2];
        n = treinoTamanho.length;
        m = testeTamanho.length;
        for (Entry<Integer, Double> entry : r1.attributes.entrySet()) {
            int i = entry.getKey();
            seq1[i] = r1.attributes.get(i);
        }
        for (Entry<Integer, Double> entry : r2.attributes.entrySet()) {
            int i = entry.getKey();
            seq2[i] = r2.attributes.get(i);
            //m = i;
        }
        warpingPath = new int[n + m][2];
        double accumulatedDistance = 0.0;

        double[][] d = new double[n][m];
        double[][] D = new double[n][m];

        for (int i = 1; i < n; i++) {
            for (int j = 1; j < m; j++) {
                d[i][j] = distanceBetween(seq1[i], seq2[j]);
            }
        }
    }
}

```

```

D[0][0] = d[0][0];

for (int i = 1; i < n; i++) {
    D[i][0] = d[i][0] + D[i - 1][0];
}

for (int j = 1; j < m; j++) {
    D[0][j] = d[0][j] + D[0][j - 1];
}

for (int i = 1; i < n; i++) {
    for (int j = 1; j < m; j++) {
        accumulatedDistance = Math.min(Math.min(D[i - 1][j], D[i - 1][j -
1]), D[i][j - 1]);
        accumulatedDistance += d[i][j];
        D[i][j] = accumulatedDistance;
    }
}
accumulatedDistance = D[n - 1][m - 1];

int i = n - 1;
int j = m - 1;
int minIndex = 1;

warpingPath[K - 1][0] = i;
warpingPath[K - 1][1] = j;

warpingDistance = accumulatedDistance / K;

this.reversePath(warpingPath);
return warpingDistance;
}

protected void reversePath(int[][] path) {
    int[][] newPath = new int[K][2];
    for (int i = 0; i < K; i++) {
        for (int j = 0; j < 2; j++) {
            newPath[i][j] = path[K - i - 1][j];
        }
    }
    warpingPath = newPath;
}

public double getDistance() {
    return warpingDistance;
}

protected double distanceBetween(double p1, double p2) {
    return (p1 - p2) * (p1 - p2);
}

```

```
    }

    protected int getIndexOfMinimum(double[] array) {
        int index = 0;
        double val = array[0];

        for (int i = 1; i < array.length; i++) {
            if (array[i] < val) {
                val = array[i];
                index = i;
            }
        }
        return index;
    }

    public String toString() {
        String retVal = "Warping Distance: " + warpingDistance + "\n";
        retVal += "Warping Path: {";
        for (int i = 0; i < K; i++) {
            retVal += "(" + warpingPath[i][0] + ", " + warpingPath[i][1] + ")";
            retVal += (i == K - 1) ? "}" : ", ";
        }
        return retVal;
    }
}
```