AN EXPERIMENTAL EVALUATION OF AUTOMATIC CLASSIFICATION OF SEQUENCES REPRESENTING SHORT CIRCUITS IN TRANSMISSION LINES

JEFFERSON MORAIM*, YOMARA PIRES†, CLAUDOMIR CARDOSO‡, ALDEBARO KLAUTAU§

*Federal University of Pará, Signal Processing Laboratory (LaPS), DEEC - CT - UFPA - Belém, Brazil, CP 8619 – 66075-110.

Emails: jmorais@ufpa.br, yomara@ufpa.br, claudomir@ufpa.br, aldebaro@ufpa.br

Abstract—This work concerns automatic classification of short circuits in transmission lines. These faults are responsible for the majority of the disturbances and cascading blackouts. Each short circuit is represented by a sequence (time-series) and both online (for each short segment) and offline (taking in account the whole sequence) classification are investigated. Results with different preprocessing (e.g., wavelets) and learning algorithms are presented, which indicate that decision trees and neural networks outperform the other methods. Another contribution of this work is to promote the adoption of a public and comprehensive labeled dataset with short circuit sequences, which allows to properly compare the algorithms and reproduce the results.

Keywords—Fault classification, sequence classification, machine learning.

1 Introduction

The electric power industry has currently a reasonably sophisticated logistics to acquire and store time series (waveforms) corresponding to power quality (PQ) [1, 2] events. A typical example is the oscillography equipments [3] that store waveforms along with additional information such as date and time, in cases where the amplitude differs from its nominal value.

More specifically in the fault analysis field, the companies are integrating their legacy supervisory control and data acquisition (SCADA) systems, which report time to the second and do not provide waveforms, with the so-called intelligent electronics devices (IEDs) such as digital fault recorder (DFRs) and digital relays, which can support sampling frequencies of tens of kHz and implement sophisticated algorithms [3]. In both fields, classification and other data mining tasks can be performed at the level of the IED (online) or at a supervisory center (postfault), which collects data from several sources.

This work investigates a particular and important class of causes of PQ events: faults in transmission lines. Studies showed that these faults were responsible for 70% of the disturbances and cascading blackouts [4, 5].

Due to the lack of freely available and standardized benchmarks, most previous publications in this area used proprietary datasets, making difficult to compare algorithms and reproduce results. Another contribution of this work is to promote UFPAFaults2, a public and comprehensive labeled dataset with short circuit sequences, which allows to properly compare the algorithms. The faults of this datasets were simulated with the Alternative Transients Program (ATP) [6]. ATP models have a long history of good reputation with respect to mimicking the actual system behavior when well tuned. Data mining techniques (pre-processing and machine learning algorithms) are then used to train and test classifiers.

Most of the literature in faults classification (see, e.g., [4]) adopts a raw or wavelet front end and neural networks as the learning algorithm. This work compares the neural networks with decision trees and other classifiers, assuming raw and wavelet front ends.

This paper is organized as follows. Section 2 presents definitions and the notation. Section 3 describes the simulation setup, including the dataset of faults and the adopted algorithms (pre-processing and learning). The simulation results are discussed in Section 4, while Section 5 presents the conclusions.

2 Classification of Time Series Representing Faults

In this work, the time series represent faults, which are basically short-circuits in transmission lines. This section defines a notation that may look abusive. However, there are many ways of representing and classifying time series, and a precise notation is necessary to avoid obscure points.

Most transmission systems use three phases: A, B and C. Hence, a short-circuit between phases A and B will be identified as “AB”. Considering the possibility of a short-circuit to “ground” (G), the task is to classify a time series into one among eleven possibilities: AG, BG, CG, AB, AC, BC, ABC, ABG, ACG, BCG, ABCG. Algorithms to solve this classification problem are used by DFRs, distance relays and other equipments (see, e.g., [3]).

The signal capturing equipments are sometimes located at both endpoints of the transmission line. Most of them are capable of digitizing voltage and current waveforms. It is assumed that each equipment has a trigger circuit that detects an anomaly and stores only the interval of interest - the fault and a pre-determined number of
samples before and after the fault. The trigger itself corresponds to a binary classification problem: “fault” or “no-fault” [7], but this interesting problem is out of the scope of the present work.

Each fault is a variable-duration multivariate time-series. The \( n \)-th fault \( X_n \) in a dataset (oscillography records, for example) is represented by a \( Q \times T_n \) matrix. A column \( x_t \) of \( X_n \), \( t = 1, \ldots, T_n \), is a multidimensional sample represented by a vector of \( Q \) elements. For this work, this work adopts \( Q = 6 \) (voltage and current of phases A, B and C) at each central sample, its two neighbors at left and right, in the case of a fault with \( T = 10 \) samples, \( X = Z \) would have dimension \( 6 \times 10 \), while \( Z \) would be a \( 30 \times 2 \) matrix. In practical systems, one can adopt \( Q = 6 \) and \( K = 198 \) [4].

Fault classification systems can be divided into two types. The first one aims at performing a decision (classification) for each frame \( F \). This is typically the goal in on-line scenarios, at the level of, e.g. a protection relay [4]. On-line fault classification must be performed on a very short time span. It is often based on a frame corresponding to half or one cycle of a sinusoidal signal of 60 or 50 Hz. Assuming, 60 Hz and a sampling frequency of \( f_s = 2 \text{ kHz} \), one cycle corresponds to \( L = 3000/60 \approx 33 \) samples. Alternatively, the decision can be made at a supervisory center in a post-fault stage. The latter case deals with matrices \( Z \) of variable dimension \( K \times N_a \), while the former with vectors \( z \) of a fixed dimension \( K \). The on-line and post-fault systems try to solve problems that can be cast as conventional classification [9] and sequence classification [10] problems, respectively.

In a conventional classification scenario, one is given a training set \( \{(z_1, y_1), \ldots, (z_M, y_M)\} \) containing \( M \) examples. Each example \((z, y)\) consists of a vector \( z \in \mathbb{R}^K \) called instance and a label \( y \in \{1, \ldots, Y\} \). A conventional classifier is a mapping \( F : \mathbb{R}^K \rightarrow \{1, \ldots, Y\} \). Some classifiers are able to provide confidence-valued scores \( f_i(z) \) for each class \( i = 1, \ldots, Y \), such as a probability distribution over \( y \). For convenience, it is assumed that all classifiers return a vector \( y \) with \( Y \) elements. If the classifier does not naturally return confidence-valued scores, the vector \( y \) is created with a unitary score for the correct class \( f_i(z) = 1 \) while the others are zero \( f_i(z) = 0, i \neq y \). The final decision is based on the max-wins rule \( F(z) = \arg \max_i f_i(z) \).

Contrasting to the on-line case, the post-fault classifier is a mapping \( G : \mathbb{R}^{K \times N_o} \rightarrow \{1, \ldots, Y\} \) and the training set \( \{(Z_1, Y_1), \ldots, (Z_M, Y_M)\} \) contains \( M \) sequences and their labels.

There are techniques for implementing \( G \) that deal directly with sequences, such as hidden Markov models (HMM) [11] and dynamic time-warping (DTW) [12]. Another alternative is the frame-based sequence classification (FBSC) architecture, in which the fault module repeatedly invokes a conventional classifier \( F \) (e.g., neural network or decision tree) to obtain the scores \( y = (f(z_1), \ldots, f_y(z)) \) for each class. To come up with the final decision, the fault module can then take in account the scores of all frames. For example, the module can calculate an accumulated score \( g_i(Z) \) for each class and then use the max-wins rule

\[
G(Z) = \arg \max_i g_i(Z),
\]

where possible alternatives are:

\[
g_i(Z) = \sum_{n=1}^{N} f_i(z_n) \tag{1}
\]

or

\[
g_i(Z) = \sum_{n=1}^{N} \log(f_i(z_n)). \tag{2}
\]

In FBSC, the accuracy of \( G(Z) \) is clearly dependent on the accuracy of the classifier \( F(z) \).

The performance of the fault classifiers can be evaluated according to their misclassification problems.
rates $E_a$ and $E_f$, for the sequence (post-fault) and frame-by-frame (on-line) modules, respectively.

One can see that there are many degrees of freedom when designing an algorithm for fault classification. The next section presents the framework adopted.

3 Simulation Setup

The simulations used the UFPAFaults2 dataset, which can be downloaded from www.laps.ufpa.br/freedatasets/UFpaFaults/. This dataset is composed by 1,000 simulated faults, which were split into two disjoint sets with 500 examples each, to be used for training and testing. An explanation about how the faults are generated with the software AmazonTP is given in [13].

3.1 Preprocessing and front end

It is beneficial to normalize the raw data such that the waveforms have amplitudes approximately in the range [-1, 1] (per unit or pu). This work used two types of normalization, which are called here prefault and allfault. In the prefault normalization, a fixed-duration interval of the signal preceding the disturbance is used to find the maximum absolute value $X_{\text{max}}$. Then, this maximum value is used as base for the conversion of each phase to pu. The allfault normalization takes in account all duration of the waveforms for getting the maximum and minimum amplitudes of each phase, and the converting to pu. Both normalizations adopt a distinct normalization factor for each of the $Q$ waveforms. For example, if one waveform has an overshoot of amplitude $X_{\text{max}}$, the allfault can convert $X_{\text{max}}$ to 1 and keep the whole waveform in the [-1, 1] range, while prefault may convert $X_{\text{max}}$ to a value much larger than 1, because its normalization factor is based on a pre-fault interval (where the amplitudes are typically close to their nominal values).

After the analog to digital conversion and preprocessing, the front end is responsible by all operations to generate the sequence that will be passed to the mining algorithms (e.g., classification and clustering). All front ends in this work assume an input sequence $X$ with $Q = 6$ raw currents and voltages at $f_s = 40$ kHz.

Wavelet is a popular representation for fault classification [14]. There are many ways of representing a sequence through wavelet coefficients. In this work, two front ends based on wavelets are investigated.

Some works in the literature use only one of the details or calculate the average power of the coefficients [15]. In constrast, the waveletconcat front end concatenates all the coefficients and organizes them in a matrix $Z$. One has to take in account that the coefficients have different sampling frequencies. For example, assuming a 3-level decomposition of a signal with $f_s = 2$ kHz, there are four signals (sequence of coefficients) for each of the $Q$ waveforms: the approximation $a$, and three details $d_1, d_2$ and $d_3$, which have sampling frequencies, given by 250, 1000, 500, 250, respectively. Therefore, instead of using a single $L$, the waveletconcat front end adopts a value $L_{\text{min}}$ for the signals with lowest $f_s$ ($a$ and $d_1$ in the previous example). The other signals use a multiple of $L_{\text{min}}$. Invoking the previous example again, $L = 3L_{\text{min}}$ and the frame lengths for $d_1$ and $d_2$ are $4L_{\text{min}}$ and $2L_{\text{min}}$, respectively.

A similar reasoning is applied to the shift $S$, which requires the definition of $S_{\text{min}}$.

The coefficients are then organized in a frame $F$ of dimension $Q \times L$, where $L = 2^k L_{\text{min}}$ for a decomposition level $k$. The number of frames is given by

$$N = 1 + \left[ \frac{(T_a - L_{\text{min}})}{S_{\text{min}}} \right],$$

where $T_a$ is the number of elements in $a$.

Another alternative for organizing the wavelet coefficients is by taking the windowed normalized total energy (average power) of each coefficient. This front end is called waveletenergy and, similarly to the waveletconcat, it has to deal with signals of different sampling frequencies. Their main distinction is that, instead of concatenating all coefficients, waveletenergy represents $X$ by its energy (or power) in frequency bands specified by the wavelet decomposition. Hence, waveletenergy loses information but can achieve a significant reduction in computational cost of the classification algorithms by decreasing the number of parameters.

3.2 Learning Algorithms

The simulations in this paper relied on Weka [9], which has many learning algorithms. Specifically, the work used decision trees (J4.8, which are a Java version of C4.5 [9]), multilayer artificial neural network (ANN) trained with backpropagation, naive Bayes and K-nearest neighbor (KNN) [9]. For KNN, instead of using the whole dataset in the test stage, the K-means clustering algorithm was adopted for finding a specified number of centroids to represent the training set [9]. This can substantially reduce the computational cost of KNN. A discriminative Gaussian Mixtures (GMM) classifier [16] was also adopted, which estimates a mixture of Gaussians for each class.

These classifiers were used for on-line fault classification experiments, where the decisions are made on a frame-by-frame basis. For post-fault classification, among the several options (HMM, DTW, etc.), this work adopts FBSC architectures by modifying the Weka code [9] to invoke the previous classifiers.
Table 1: Approximate computational cost for some classifiers, where \textit{sigm} is a sigmoid function and \textit{mac} is a multiply and add operation, executed in one cycle in modern DSP chips.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>J4.8</td>
<td>$\log_2 p_b$ if-else</td>
</tr>
<tr>
<td>ANN</td>
<td>$(Y + p_h) \text{ sigm}, (K + Y)p_h \text{ mac}$</td>
</tr>
<tr>
<td>KNN</td>
<td>$p_m K \text{ mac}$</td>
</tr>
<tr>
<td>GMM</td>
<td>$Y p_y K \text{ mac}$</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>$Y K \text{ mac}$</td>
</tr>
</tbody>
</table>

It is interesting to compare the computational cost of the classifiers during the test stage. The training stage can be done offline and is typically less important. The cost depends on the complexity of each classifier and can be roughly estimated as follows. A binary decision tree as J4.8 with $p_b$ internal nodes requires $\log_2(p_b)$ comparisons to reach a leaf. Assuming ANNs with one hidden layer of $p_h$ nodes, each node computes an internal product between vectors of dimension $K + 1$ (input dimension plus bias) and calls a sigmoid function. Similar work is done by each of the $Y$ nodes of the output layer, but with vectors of dimension $p_h$. A KNN classifier using $p_m$ stored vectors (training instances or centroids calculated by clustering) and the Euclidean distance, requires computing $p_m$ internal products or, alternatively, $p_m$ squared norms of vectors of dimension $K$. Assuming each of the $Y$ classes being modeled by a mixture with $p_y$ Gaussians, the GMM classifier requires $Y p_y K$ Euclidean distances for calculating log-likelihoods. When the Naive Bayes classifier uses a one-dimensional Gaussian for each element of the input $z$ (as in this work), its cost is equivalent to a GMM with one Gaussian per class. This rough estimate of the total cost at the test stage of these classifiers is summarized in Table 1.

To provide a single cost estimate, the table suggested in [17] was used to weight the different operations (arithmetic, logical, etc.). These weight obviously depend very much on the computing platform but they are helpful to provide a first approximation.

3.3 Parameter selection

Often, the best performance on a particular dataset can only be achieved by tedious parameter tuning. This is a computational intensive approach, but avoids tuning the parameters by repeatedly evaluating the classifier using the test set. A popular strategy is to perform this parameter (or model) selection using cross-validation.

It should be noted that, conventionally, the examples are assumed to be independently and identically distributed (iid) “samples” from an unknown but fixed distribution $P(z,y)$. However, this assumption is invalid when training and test datasets of fault classification experiments can have vectors $z$ extracted from the same sequence. This fact is important in practice because when performing model selection based on, e.g. cross-validation, the procedure can lead to overfitting because vectors that were extracted from the same waveform have relatively high similarity among them.

Because of that, automatic model selection used a validation file disjoint with respect to the train and test files. Combinations of possible parameters were specified as a \textit{grid} and searched exhaustively.

4 Experimental Results

In the pre-processing stage, the 40 kHz waveforms were decimated by 20 to create sequences with $f_s = 2$ kHz.

Results for the three different front ends in Table 2 are discussed. The first front end, which is called \textit{raw}, adopts $L = 9$ raw samples (central, four at the left and four at the right), such that $K = 54$. There is no overlap ($S = L$). The two wavelet front ends used a Daubechies 4 (db4) [8] with a 3-level decomposition. Hence, for each of the $Q = 6$ waveforms, the wavelet decomposition generated four signals. The waveletconcat adopted $L_{\text{min}} = 4$ and $S_{\text{min}} = 2$, while waveletenergy used $L_{\text{min}} = S_{\text{min}} = 1$. For example, Assuming a $6 \times 5000$ matrix $X$ (already at $f_s = 2$ kHz), waveletconcat generates a sequence $Z$ with $K = 192$ and $N = 311$ frames. Table 3 indicates the parameters of the classifiers obtained by the automatic model selection procedure.

Figure 1 shows a comparison between the two normalization methods adopted in this work. For both, the best results were obtained by the ANN and J4.8 classifiers. The large difference for the Naive Bayes classifiers requires further analysis.

Figure 2 shows the results for the two wavelet front ends (the results for the raw are repeated for convenience). These two did not outperform the best result obtained with the front end \textit{raw}. One should notice, however, that there are many degrees of freedom when designing a front end based on the wavelet transform, and these should be seen as baseline results.

Figure 3 indicates the estimated computational cost of some classifiers. Model selection
chooses different classifiers when the normalization strategy changes. An exception is the Naive Bayes classifier, which does not have such parameters and presents the same cost.

Some FBSC post-fault classifiers $G$ were designed using Eq. (1) and the max-wins rule. Figure 4 shows $E_f - E_s$, the absolute reduction in error rate when comparing sequence classification and the corresponding conventional classification. For example, the GMM presents $E_f = 51.3\%$ and $E_s = 31.6\%$, which leads to a difference of 19.7%.

Figure 5 shows a comparison of the robustness of the J4.8 and ANN classifiers to the addition of white Gaussian noise (AWGN) to the waveforms. Both classifiers were trained with waveforms not contaminated by noise and tested under a condition of a signal to noise ratio of 30 dB, i.e. a forced mismatch condition between train and test. It can be seen that in this case, J4.8 was slightly less robust to noise than the ANN.

5 Conclusions

This paper presented a thorough description of the issues related to the design of fault classification modules for power electric systems. The solutions to this problem involve digital signal processing and machine learning algorithms. Consequently, there are many degrees of freedom when designing a classifier. For example, the wavelets front ends would probably benefit from finer tuning.

The experimental results indicated that neural networks and decision trees outperformed the other classifiers. Decision trees seem particularly interesting when one is trying to minimize the computational cost, such as in the development of embedded devices. Neural networks achieved a better accuracy and improved robustness.

The post-fault classification deserves more investigation. The FBSC architecture is just one among many options. It is interesting that a GMM classifier, which outputs log-likelihoods, had a large discrepancy between $E_s$ and $E_f$, while the difference was much smaller for ANN. This is another topic that deserves further investigation.
Figure 4: Difference $E_f - E_s$ between the error rates for frame-by-frame and sequence classification using prefault normalization and the raw front end.

Figure 5: Difference between datasets with noise and without noise (the first set of results is the same as in Fig. 1).

Acknowledgements

Thanks to Eletronorte and Prof. Marcus Nunes (GSEI/UFPA) for sharing the ATP models.

References


