MACHINE LEARNING FOR SMART GRID EVENT DETECTION

By

DUC THE NGUYEN

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

WASHINGTON STATE UNIVERSITY
School of Engineering and Computer Science, Vancouver

MAY 2016
To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of DUC THE NGUYEN find it satisfactory and recommend that it be accepted.

Scott Wallace, Ph.D., Chair

Xinghui Zhao, Ph.D.

Sarah Mocas, Ph.D.
ACKNOWLEDGMENTS

First, I would like to express appreciation to Professor Scott Wallace for advising me in the Master of Science program at Washington State University Vancouver, and Professor Xinghui Zhao for her supporting and providing many useful suggestions and discussions. I am happy to be a member of the research team in Bonneville Power Administration’s technology innovation project managed by them. I would like to thank my research colleagues Richard Theodore Barella who showed me useful features for classifying the collected data, Eric Bryan Klinginsmith and Kuei-Ti Lu for communications and discussions.

I would also like to thank the DoE / the Bonneville Power Administration for funding and supporting our research group through the Technology Innovation Program (TIP# 319), OregonBest for naming me as recipient for their scholarship in 2014, and William Bowen - my brother-in-law - for carefully and patiently providing proof-reading for my thesis manuscript.

Finally, I would like to thank to my family and wife for encouraging and helping me to manage my time and to balance work with fun.
Abstract

by Duc The Nguyen, M.S.
Washington State University
May 2016

Chair: Scott Wallace

Synchrophasor technology with Phasor Measurement Units (PMUs) deployment on electrical transmission lines has enabled real-time wide-area monitoring and a new opportunity to enhance situational awareness on the power grid system. With the high sample rate, the technology has presented a new data stream for grid operators to examine details of the dynamic behaviors of electrical systems. The technology also has brought a new attention to the problem of how to leverage the information from large-scale data streams generated by PMUs to improve situational awareness in control rooms. In this thesis, we introduce an approach to employ machine learning techniques for characterizing and classifying line events from the data streams. These events include Single Line to Ground, Line to Line, and Three Phase faults.

Our work examines Bonneville Power Administrations historic synchrophasor data recorded between October 2012 and September 2013 and proposes an approach for wide-area line events detection and eventually constructs a multiple stage cascade classifier for detecting line events for use with its current PMU installation. The performance of our classifier is
estimated and compared with seven supervised learning algorithms that typically perform well with large-scale data sets. We show that the learning model generated by our classifier algorithm outperformed the examined algorithms with respect to producing very low false alarms and highly accurate outcome predictions.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>iii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>iv</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>ix</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>xiii</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Research Contribution</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Thesis Outline</td>
<td>4</td>
</tr>
<tr>
<td>2 Background</td>
<td>6</td>
</tr>
<tr>
<td>2.1 BPA synchrophasor</td>
<td>6</td>
</tr>
</tbody>
</table>
2.2 Statistical performance measurement .................................. 8
  2.2.1 Confusion matrix ...................................................... 8
  2.2.2 Measurement scores .................................................. 9
2.3 Hand-built classification rules .......................................... 10
  2.3.1 Expert decision tree .................................................. 10
  2.3.2 Expert derived features .............................................. 11

3 Related Work .............................................................. 14

4 Machine Learning for Smart Grid Event Detection ................. 18
  4.1 Training and Testing data ............................................ 18
    4.1.1 Line event examples ............................................. 19
    4.1.2 Normal operation examples ................................... 22
  4.2 Experiment objective .................................................. 23
  4.3 Developed features .................................................... 23
    4.3.1 Relative deviation features ................................... 23
    4.3.2 Relative phase angle features ................................ 25
  4.4 Cascade of SVMs ....................................................... 27
    4.4.1 Three-stage classifier ........................................... 28
    4.4.2 Five-stage classifier ............................................. 39

5 Evaluation .................................................................... 44
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Typical confusion matrix</td>
<td>9</td>
</tr>
<tr>
<td>5.1</td>
<td>Testing dataset performance of the proposed classifiers</td>
<td>46</td>
</tr>
<tr>
<td>5.2</td>
<td>Testing dataset performance of new 5-stage proposed classifiers</td>
<td>47</td>
</tr>
<tr>
<td>5.3</td>
<td>Testing dataset performance of studied classifiers. The boldface marker at a score indicates the best score. The * marker at a score indicates the score closest to the best.</td>
<td>50</td>
</tr>
<tr>
<td>5.4</td>
<td>False positive found by corroborated classifications</td>
<td>62</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>PMU Topology: Nodes indicate substation sites (shaded nodes indicate PMU locations); edges indicate transmission lines instrumented with a PMU.</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Per-Unit Voltage Magnitudes and Sag Voltage During a Fault</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>Expert decision rules. The rounded rectangle nodes are the decision nodes and cycle nodes are leaf nodes. This is a binary split tree model in which the decision nodes has two discrete branches. Given an input, at each decision nodes, a test function is applied and the result of the test directs the outcome by taking the either branch. If the test functions return True, the right moving branches are taken. Otherwise, the downward branch are taken.</td>
<td>12</td>
</tr>
<tr>
<td>4.1</td>
<td>Typical voltage signature during a line event</td>
<td>21</td>
</tr>
<tr>
<td>4.2</td>
<td>Typical voltage signature during a line event and its impact on service buses. The vertical line indicates the moment at which the number of signal buses experiencing the largest deviations is the most.</td>
<td>24</td>
</tr>
</tbody>
</table>
4.3 Voltage sag types - ABC classification ................................. 25

4.4 Phasor diagrams: normal operation (left) and fault condition (right) ........ 26

4.5 Decision Tree Improved using Relative Deviation, a Novel Feature (sag: p.u.
voltages of each phase in increasing order; dev: absolute value voltage changes
from steady state measured in p.u., arranged in decreasing order.) ............ 28

4.6 Three-stage classifier .......................................................... 29

4.7 The splitting/cross-validation approach for model selection. The process takes
training dataset and one SVM setting to yield training, testing, and cross-
validation scores. Training scores are yielded by using the same training
datasets for training and testing purposes. The training dataset is then split
into cross-validation and testing sets. Testing scores are yielded by using
cross-validation and testing sets for training and testing purposes respectively.
Cross-validation scores are the average of the scores from cross-validation
stratification ................................................................. 31

4.8 Feature space of sorted phase and angle change from across PMUs. The left
plot displays the measurements of entire line events across power grid. The
right plot zooms in the space near (1.0, 1.0) .................................. 33

4.9 Performance metrics of feature set: vsags1 and angles1. The subplots at the
left side demonstrate performance metrics of polynomial kernel. The subplots
at the right side demonstrate performance metrics of RBF kernel ............... 34
4.10 Performance metrics of feature set: $vsags_1$ and $angles_1$ with the polynomial kernel. The left subplot illustrates the change of the error rate when the larger $C$ is set, and the right subplot shows the sensitive and specificity in the context of tuning process.

4.11 Feature space of sorted phase deviation after removing no-fault examples

4.12 Performance metrics of the feature set: $rel\_phase_{31}$ and $rel\_phase_{32}$. The two subplots at the left side demonstrate performance metrics of polynomial kernel. Meanwhile, the two subplots at the right side demonstrate performance metrics of RBF kernel.

4.13 Feature space of sorted phase deviation after removing nofault and 3P examples

4.14 The error rates on two kernels: polynomial and RBF by using the sets of features (a) $angles_1$ and $rel\_phase_{21}$, (b) $rel\_phase_{31}$ and $rel\_phase_{32}$, and (c) $vsags_1$ and $vsags_2$

4.15 Performance metrics of feature set: $angles_2$ and $rel\_phase_{21}$. The subplots at left sides demonstrate performance metrics of polynomial kernel. The subplots at right sides demonstrate performance metrics of RBF kernel.

4.16 Performance metrics of feature set: $angles_2$ and $rel\_phase_{21}$ with the RBF kernel. The left subplot illustrates the change of the error rate when the larger $C$ is set, and the right subplot shows the sensitive and specificity in the context of tuning process.
4.17 Unequal distribution of \( vsags_1 \) in the range between 0.98 and 1.0. The distribution of \( vsags_1 \) in different fault types SLG, LL and 3P are showed respectively.

4.18 Five-stage classifier

5.1 Improved-five-stage Cascade of SVMs

5.2 ROC curve for proposed classifiers

5.3 The tradeoff between true positive rate and corroborated fault classification. The false positive rates are annotated at the points in which the rates are changed as the threshold value is increased on the y-axis.

5.4 A steady state problem during classifying.
Chapter 1

Introduction

1.1 Motivation

Synchrophasors are an emerging smart grid technology enabling grid operators to monitor the health and global operation of a power grid in real time. A core component of the technology is the Phasor Management Unit (PMUs) which capture time-synchronized measurements. The measurements represent the voltage magnitude, current, phase angle and frequency found during electricity transmission across a geographically distributed area. The PMUs’ phase-clock is associated with Global Positioning System Satellites to reference a precise time for accuracy. PMUs have the ability to record synchronized measurements at a rate of 10-60 samples per second for a 50/60Hz AC waveform that are much higher than conventional monitoring technologies such as supervisory control and data acquisition (SCADA) which measure once every two to four seconds [1, 2]. With the high sampling rate, synchrophasor technology provides an accurate and near real-time insight to the power grid stability and line event experiences. Thus, it advances the situational awareness of the system. The advancements of the synchrophasor technology have drawn significant attention from academia and the power grid industry since its appearance. A significant amount of
A synchrophasor project is investigated in order to: 1) enhance the efficiency and reliability of the power grid system; 2) increase response-based automatic control to draw attention to wide-area awareness situations; 3) improve system planning and system analysis by conducting power system performance base-lining, model validation, and event analysis [1, 3].

Although PMUs are a potential technology to monitor the concerns in operating electric transmission systems, they also present challenges for data processing with traditional workflow due to the large amount of data [4]. With a high sampling rate of 60Hz, a single PMU generates 5,184,000 samples per day. Each sample can be stored with 146 bytes composed of 8 bytes time stamp; 2 bytes status flag; 8 bytes for frequency and rate of frequency change; and 16 phasors with 8 bytes floating point value for phasor magnitude and another for phase angle. These combine to approximately 721 MB for one day of data per signal. For wide-area management systems, the amount of data can add up very quickly. For instance, our accessed synchrophasor data was recorded from The Bonneville Power Administration (BPA) with 44 installed PMUs throughout the Pacific Northwest. There is approximately 1 TB per month to be collected in the system. Such data streams continue to expand over time and become too large for humans to process without computer assistance. It will be increasingly difficult for grid operators to leverage the latent knowledge of these vast data streams and transpose them into useful information for making informed decisions in control rooms.

In the emerging era of big data, machine learning techniques have become the de facto standard for processing and analyzing large-scale data streams with little human involvement. We believe that employing the techniques can leverage the latent knowledge in these vast PMU data streams for increasing the smart grid reliability and assisting grid operators in smart grid management [5].
1.2 Research Contribution

In this thesis, we empirically examine the conditions of line events during the occurrence of Single Line to Ground (SLG), Line to Line (LL), and Three Phase (3P) faults. When the power grid experiences a fault at the fault location, every service bus or line will experience certain variations, which could result in voltage sags and/or different standing phase angles of phasors in the service buses. The event ends with the trip of circuit breakers to bring the power grid system back to normal operation. Due to this dynamic behaviour of the power grid system, we have no pre-defined thresholds or conditions for distinguishing line events from normal operation conditions. We believe that machine learning techniques can rely on statistical models to generalize the changes caused by line events and identify the events automatically.

We focus on automatically detecting and classifying line events in archived PMU data using machine learning techniques. Our experiments are conducted on archival data from a large, active power transmission system spanning the Pacific Northwest region of the United State of America. Our works contribute as follows: (i) to propose the appropriate approach for automatically generating a learning model which performs a wide-area line events detection, (ii) to exploit a variety of machine learning algorithms for our approach, (iii) to propose a multiple-stage cascade classifier for modeling our line events and (iv) to perform a detailed analysis of false alarms in large-scale datasets to explore multiple methods for optimizing false alarms in large-scale detection.

Many of the results presented in this thesis have been presented individually, often using slightly different data sets that used for this examination:

- “Finding Needles in a Haystack: Line Event Detection on Smart Grid PMU Data Streams” [6] proposes a five-stage classifier which improves upon the experts hand-built classification rules and a decision tree generated by the J48 learning algorithm.
We subsequently evaluate the error rate of these learning models and explores methods for reducing false positives on a stimulated operating environment.

- “Big Data Analytics on Smart Grid: Mining PMU Data for Event and Anomaly Detection” [7] presents our examination in detecting line events, and anomalies from BPAs synchrophasor data. A three-stage classifier for identifying line events is proposed for this subject. The purpose of the study is to enhance the traditional workflow of analyzing the archival PMU data to identify events.

- “Smart Grid Line Event Classification Using Supervised Learning Over PMU Data Streams” [8] demonstrates that machine learning techniques can be used to enhance the identification of line events when characterized the voltage sags at a distance form fault locations. The study also presents the novel features and its efficiency in separating three fault types. Almost our proposed and examined classifiers in the thesis are derived from these features to construct the learning models.

- “A Sensitivity Study of PMU-Based Fault Detection on Smart Grid” [9] analyzes the quantitative impacts how the distance between a fault location and the PMUs reflects the detection process given by the domain exports classification rules. Relying on the experimental results, the study presents a guide line for an efficient PMU deployment plan.

1.3 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 introduces the archived PMU data from Pacific Northwest provided by BPA, the concept of machine learning techniques and the measurement for comprehensive machine leaning detecting models’ performance. Chapter 3 outlines the previous studies that employ machine learning approach for the line
event detecting problem. Chapter 4 delves into our method and algorithms employed in the event detection. The experiment setup and evaluation results are explained in the Chapter 5. Finally, in Chapter 6, we summarize and conclude our work.
Chapter 2

Background

This section presents the dataset and its associated line events annotation provided by BPA domain experts. Next, we introduce statical performance metrics that are used to convey learning models’ performance. The hand-built classification rules proposed by our domain expert are introduced in turn.

2.1 BPA synchrophasor

The thesis work is conducted with two folds of BPA synchrophasor data. One of two folds are collected in Pacific Northwest area in the United States from October 17, 2012 to September 16, 2013. The another one are in the same setting recorded from October 22, 2013 to October 10, 2014. At the time there are 44 PMUs operating at 34 substations measuring voltage magnitude, current, and frequency data. The synchrophasor data contains both 500KV and 230KV buses. Figure 2.1 gives a general idea about network topology and how PMUs deployments are placed in BPA smart grid. Note that in the figure site names are replaced with numbers at the request of BPA. The figure also does not show transmission lines which are not monitored by any PMU. The installed PMUs record at high frequency 60Hz. This
means there are 60 measurements to be written by a PMU within a second. We use the 2013 dataset for generating and constructing a learning model at the development phrase. Meanwhile, the 2014 data is used to perform large-scale testing which simulates the real-time environment.

There are 110 line events documented by BPA’s domain experts including SLG, LL and 3P faults associated to this 2013 dataset. Each line event is documented with several fields: staring timestamp, ending timestamp, comment, and duration of line event in minutes. Of these 110 line events there are 10 line events found duplicated. After our manual examination evaluates and eliminates these 10 line events, we collect 90 SLG line events, 6 LL line events and 4 3P line events for our study. Figure 2.2 illustrates a typical LL voltage sag line event recorded by a PMU. The sag voltage of a signal in the plot is localized by searching the largest dip after the starting timestamp of a line event. Note that the voltage magnitudes of three phases (A, B and C) in the figure are normalized by their steady state values into the so called per unit system. Additionally, there are 800 minutes data which are not associated with any line events.

Figure 2.1: PMU Topology: Nodes indicate substation sites (shaded nodes indicate PMU locations); edges indicate transmission lines instrumented with a PMU.
Figure 2.2: Per-Unit Voltage Magnitudes and Sag Voltage During a Fault

2.2 Statistical performance measurement

2.2.1 Confusion matrix

In predictive analytics, a confusion matrix contains information to survey the actual and predicted classification results performed by a classifier. It provides more detail analysis than the proportion of correct predictions. For the rest of the thesis the confusion matrix is provided in a table layout with two or more rows/columns. As illustrated in Table 2.1, each column represents the number of examples in a predicted class, while each row represents the number of examples in actual/labeled class.

The numbers in the confusion matrix have the following meaning in the context of our study:

- True Positive (TP): the number of the positive instances that were correctly classified into the positive class.
<table>
<thead>
<tr>
<th>Actual</th>
<th>Positive class</th>
<th>Predicted</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TP</td>
<td>FN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FP</td>
<td>TN</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: Typical confusion matrix

- **False Negative (FN):** the number of the *positive* instances that were *incorrectly classified* into the negative class.
- **False Positive (FP):** the number of the *negative* instances that were *incorrectly classified* into the positive class.
- **True Positive (TN):** the number of the *negative* instances that were *correctly classified* into the negative class.

### 2.2.2 Measurement scores

- **Accuracy:** the rate of correct predictions over all population of a data set. It gives intuitive performance made by the predictive model and is usually estimated by using an independent test set that was not used during the training process. It is determined using the equation:

$$ Accuracy = \frac{TP + TN}{TP + FN + FP + TN} $$

- **Hamming loss [10]:** the rate of incorrect predictions over all population of a data set. It is determined using the equation:

$$ Hamming\ loss = \frac{FN + FP}{TP + FN + FP + TN} $$

- **Sensitive score:** the rate of correct *positive* predictions over the proportion of positives.
It indicates the probability of a positive example to be classified when it presents. It is also called true positive rate or recall. It is determined using the equation:

\[
\text{Sensitivity} = \frac{TP}{TP + FN}
\]

- Specificity: the rate of correct negative predictions over the proportion of negatives. In the same fashion, it indicates the probability of a negative example to be classified when it presents. It is also called true negative rate. It is determined using the equation:

\[
\text{Specificity} = \frac{TN}{FP + TN}
\]

- Fallout: the rate of negatives that are incorrectly classified as positives. It is also called the false positive rate. It is determined using the equation:

\[
\text{Fallout} = \frac{FP}{FP + TN} = 1 - \text{Specificity}
\]

2.3 Hand-built classification rules

2.3.1 Expert decision tree

In 2014, our domain expert introduced a simple set of decision rules for identifying line faults from their voltage sags (A,B,C) on the same synchrophasor data [8]. The expert rules are the threshold-based rules manually constructed by examining theoretical and statistical data measured at the fault locations which experienced the largest voltage deviation [11, 12]. We also verify the fault location of each of the 57 faults given by BPAs dispatch notes with the service bus experiencing the most voltage deviation. The PMU at the service bus with the most voltage deviation is always a PMU directly connected to the line experiencing the
fault (e.g., site 38 or 27 from the example above) when PMUs on both sides of the line are operational. The voltage sag values are in the per-unit system of measurement which is to say that the voltages at a faulted cycle time are normalized with respect their steady state values. The steady state value is calculated as a median of 10-12 cycles backward from the faulted cycle time. Classifying a voltage signal at a moment of time will result in one of the following predictions:

- **SLG:** is returned if there is an existing phase with the sagged voltage which is less than or equal to 0.95 while the other two phases have sagged voltages greater than 0.93.
- **LL:** is returned if the smallest and second smallest sagged voltages are in range 0.35 and 0.93 while the other phase has sagged voltage greater than 0.88.
- **3P:** is returned if the sagged voltages of the three phases are in range 0.0 and 0.8
- **No fault:** is returned if the voltage signals are not falling to the three cases above.

Figure 2.3 visualizes the arrangement of the test functions to classify the faults. Every decision node has two discrete branches and the test functions examine one of the three with respect to a fixed threshold. The test functions are applied sequentially from the very first decision nodes of the tree until a leaf node is hit at which point the fault types written in the leaf constitute the output.

### 2.3.2 Expert derived features

Three voltage sag values mentioned above are expert derived features. The values are the sorted normalized values of the voltage magnitudes with respect steady state windows. Thus, \( \text{vsag}_1 \) corresponds to the voltage phase with the most significant sag (lowest voltage) while \( \text{vsag}_3 \) corresponds to the voltage phase that is closest to steady state. Given the time \( t \), the steady state voltage for each individual phase is computed as follows:
Figure 2.3: Expert decision rules. The rounded rectangle nodes are the decision nodes and cycle nodes are leaf nodes. This is a binary split tree model in which the decision nodes has two discrete branches. Given an input, at each decision nodes, a test function is applied and the result of the test directs the outcome by taking the either branch. If the test functions return True, the right moving branches are taken. Otherwise, the downward branch are taken.

**ALGORITHM 1:** SteadyStateVoltage(window, istop, i, bus, phase)

```
nominal = nominal voltage of the given signal;
while i! = istop do
    let sub_window be a sliced window[i – 20 : i];
    calculate the median voltage magnitude (median) over the sub_window;
    get the minimum voltage magnitude (min) over the sub_window;
    get mostLeftEdge and mostRightEdge over the sub_window;
    let edgemin be 0.99 * median if min ≥ 0.9 * median AND
    mostLeftEdge ≥ edgemin AND mostRightEdge ≥ edgemin then
        if median ≥ 0.8 * nominal then
            return median (steady state voltage);
        else
            Raise steady state does not exist;
        end
    else
        endtime = endtime – 1cycle;
    end
end
return Raise steady state does not exist;
```
Using the steady state voltage computation, we can then calculate the per-phase voltage sags at interest measurement.

\[ sag_i = \frac{V_i}{V_{ssi}} \]  \hspace{1cm} (2.1)

where \( V_i \) (i = Phase A, B, or C) is the voltage magnitude of phase i at measurement point, and \( V_{ssi} \) is the steady state bus voltage before the measurement point.
Chapter 3

Related Work

Synchrophasors generated by PMUs are time-aligned accurate measurements of smart grid systems to make system-monitoring tasks more feasible for operators. Advancements in synchrophasor technology have been recognized since the last decade. The PMU deployment across North America has been significantly increased. In 2009, there were roughly 200 PMUs installed across the electric grid in the U.S. [13], and this number has recently been increased to 1,700 [14]. A huge amount of funding has been used for the installation of PMUs and development of applications to leverage the latent synchrophasor data for increasing power grid intelligence. However, the increasing number of PMUs and their high frequency sampling rates present challenges for applying the traditional workflow of data processing and analytics.

In the last decade machine learning techniques have become prominent for processing and analyzing large-scale data in several fields. Replacing our traditional programming algorithms, the new techniques expect results that are driven by dynamic data and hidden information underlying the data. Its application areas have developed significantly. In particular, Internet-related technologies such as search engines, recommendation systems and email spam filters are utilizing machine learning. The general goal of machine learning tech-
niques is to discover knowledge based on reliable recorded data and generate a predictive model. For example, in market supply chains, customers come to the store and buy something. We know that the things bought by customers are not completely random. If we can construct a good model for describing customer buying behaviors based on previous store sales records, store managers can more accurately oversee and control inventory.

Since the power grid system is also dynamic, it presents interesting research challenges. There is a significant amount of works and studies leveraging these techniques to mine synchrophasor data for detecting and characterizing signatures of line events. Jian et al. propose an online method for fault detection and localization using SDFT (smart DFT) [15]. Liu et al. use Frequency Domain Decomposition for detecting oscillation [16]. Anzai et al. use auto-regressive models to estimate the margin of real time voltage stability. This proposed approach can be used to detect voltage collapse events. Kazami et al. introduce a multivariate regression model to localize fault locations [17]. Instead of studying current and voltage magnitudes, Tate et al. leverage phase angle measurements in detecting outages [18].

Machine learning techniques also employ another two general categories: unsupervised and supervised learning. These types of learning explore the input space and build learning models based on implemented optimization functions for certain data [19]. In unsupervised learning, the data is classified to an output whose valid classifications are not provided. The most common method of unsupervised learning is clustering or grouping data based on regularities in the input. Antoine et al. propose to identify causes for inter-area oscillation by clustering a number of parameters including mode frequency, the voltage angle differences between areas and the mode shapes [20]. It has been shown that by clustering these parameters, changes in inter-area oscillation can be explained. However, since there are no valid classifications to be provided, the clustering results usually require extra steps of comprehensive interpretation and validation given by experts.

In contrast to unsupervised learning, supervised learning methods require experts’ sup-
port to label the input data [19]. After training, models obtained using this method can perform the automatic identification without domain knowledge from experts. Zhang et al. propose a classification method for finding fault locations based on pattern recognition [21]. The key idea is to distinguish a class from irrelevant data elements using linear discriminant analysis. The classification is carried out based on two types of features: nodal voltage, and negative sequence voltage. Similar classification techniques are used to detect voltage collapse [22] and disturbances [13] in power systems. Specifically, Diao et al. develop and train a decision tree using PMU data to assess voltage security [22]. Ray et al. build Support Vector Machines and decision tree classifiers based on a set of optimal features selected using a genetic algorithm [13]. Support Vector Machine-based classifiers can also be used to identify fault locations [23], and predict post-fault transient stability [24]. Although classification approaches are effective in grouping a data element into one of several pre-defined categories, they are not sufficient to make continuous valued decisions. In this case, regression trees can be used. For example, a regression tree based approach is used for predicting stability margins, i.e., how far the system is away from a disturbance [25]. Regression trees have also been used for on-line dynamic security assessment under the impact of missing PMU data [26]. Al Karim et al. proposed a three-stage model for fault analysis [27]. The model combines three machine learning algorithms in the following order: K-nearest neighbor, Naive Bayes, and K-means for fault pattern recognition. Although there are a number of proposed works to analyze synchrophasor data for event detection and prediction, the application of real-world synchrophasor data in studies is very limited. Gomez et al. use data collected form a large-scale Wide Area Measurement System (WAMS) [24]. They apply SVM based algorithms to model practical power systems. However, this study is restricted to 15 PMUs. All other approaches mentioned in this section are based on simulations. Since simulated data can contain some uncertainty due to the condition of data generation, the results obtained may provide wrong or misleading conclusions [28].
The experiment conducted in this thesis benefits greatly from the archived data recorded in Northwest BPA’s operational grid. In our work, we use supervised machine learning techniques to construct an automatic event classification for the dataset. Through the study, we also explore several learning algorithms using Weka [29] to search for tailored algorithms. We propose a learning algorithm and evaluate it with the training and testing set extracted from BPA data in the next step. Additionally, to the best of our knowledge, there is no previous effort focusing on reducing false alarms in classifying PMU data. Since the line events happen rarely (on the order of 100 per month), the reducing false alarms becomes critical. Finally, in the later step, we perform a detailed analysis of false alarms and explore multiple methods for reducing false alarms of the implemented learning model.
Chapter 4

Machine Learning for Smart Grid

Event Detection

4.1 Training and Testing data

Machine learning is a field of programming in which algorithms process data to construct mathematical models based on the statistical theories. It is critical to come up with a good input representation, which also provides the features that will be used to represent the concepts that are learned from the data [30]. Training data usually contains a list of examples, represented as feature vectors, from real world data. For training their decision tree, Diao et al. use a number of operational conditions retrieved from one-day PMU measurements to represent examples and label the examples either Secure or Insecure based on the detailed voltage security analysis collected the next day [22]. During simulation, the decision tree model also tweaks its threshold value to improve prediction outcome each day by observing the change of measurements and network topology. Ray et al. propose the data pre-processing step through converting vectors and selecting the optimal features by genetic algorithm [13]. The study attempts to classify power quality disturbances in dis-
tributed generation systems. The model is later tested in a simulation environment. Dahal et al propose to utilize an adaptive Hoeffding tree for classifying the PMUs measurement as either stable or instable. The tree accepts the PMUs measurements from each time-stamped data point to update sufficient statistic in leaf node of the tree and yield predictions. The examples for training set are collected from the measurements generated by two PMUs in a simulated environment.

Our study focuses on the voltage sagged problem caused by a line event. We attempt to generate a learner to determine the voltage sags in real time and classify moments in time as either line event or normal. The study leverages archival PMU data recorded by BPA between October 2012 and September 2013. During this period, the BPA recorded 100 line events. Among them, only fifty-seven included log notes sufficient to determine a specific fault type (e.g. single-line-to-ground, line-to-line or three-phase). The remaining events are plotted and manually inspected to verify their voltage signatures and infer their fault time. Our examples for training and testing sets are extracted from these line events and normal operation data. Since the line events can occur at any voltage level, we are interested in generating a learning model that can assess sagged voltage events. Then, our data collection method takes advantage of wide-area PMU measurement across the power grid system, instead of using only local measurements at fault locations proposed in [31].

4.1.1 Line event examples

For line event examples, we begin measuring total voltage deviation, a metric described in [31] and useful for identifying fault signatures. The voltage deviation is defined as follows:

\[
\Delta V = \sqrt{\frac{(1 - V_A)^2 + (1 - V_B)^2(1 - V_C)^2}{3}}
\]
where

\[ V_A, V_B, V_C \]

represents the voltage of phase [ABC] in the *per unit system* at the point of interest. The detailed calculation is described in Algorithm 2.1 of Section 2.3.2, while 1 stands for the steady state voltage of phase A in the *per unit system*. Every described line event includes the exact times OutDatetime and InDatetime which represent the starting and ending time of the event. Although the fault duration can be recorded as several hours, starting time tends to be recorded very close to line event occurrence. Empirically, fault events that occur at a bus or transmission line also affect connected buses. Because of this connectivity, typical power system disturbances can spread to large areas and affect many service buses, so it is valuable for the system is able to detect these events and react accordingly. Our collection data strategy attempts to make three observations as follows. First, an example is collected at a moment of time at which its signal bus experiences significant voltage sags - the largest voltage deviation. As the result, the example is easily distinguished from normal operations. Second, for wide-area line event detection, the data contains the measurements with different voltage levels collected at signal buses across the power grid. Third, we choose a moment of time: 1) being close to the starting time of the line event, and 2) being voted by the most PMUs as the largest voltage deviation.

With this strategy, we collect fault examples for 100 line event signatures given by BPA domain experts. Using the method outlined in Liang et al [31] on these line events, we find the PMU whose value experiences the largest deviation from its steady state value, where steady state is measured by the median voltage in the 20 cycles prior to the fault. We then vote on the moment in time at which the most service buses experience the largest voltage deviation. The PMUs measurements at the selected time are then sampled across the grid. It is worth noting that this procedure only looks at the valid signal buses. A valid signal bus
is the bus at which the voltage magnitude of phases [ABC] can be measured and hold non-zero voltage magnitudes. From time to time, we may fail to record these bus measurements because PMUs connected to the bus may be offline or some signals from a specific PMU may be missing in our dataset. When this happens, these signal signatures are not considered during this step.

Fault examples in our test set are produced from the same 100 line events, but at the cycle preceding the moment of time used to generate the training instances. Figure 4.1 illustrates a typical training example point (square) and testing example point (triangle). Although these examples are not entirely independent of the training examples, we believe that they can serve as a reasonable proxy for new archival data with unseen line events. Indeed, they may even be a more challenging test than new events since the selection methodology ensures that they will typically be smaller deviations from the steady state than the examples used in training.

This approach yields 7,125 SLG examples, 474 LL examples, and 267 3P examples for both every training and testing sets.
4.1.2 Normal operation examples

Our learner is able to distinguish normal operation from line event examples. There are roughly 800 minutes annotated as normal operations by BPA. In total, these minutes contain 800x3600 individual data examples for each signal present. In contrast, line events happen rarely on a transmission system. Although we can add these normal data examples into our collective training and testing set, the datasets become imbalanced, and make it difficult to model the general learning input space of line events. The performance metrics can mislead us because the underlying distribution of major classes will dominate the results. As we might have guessed, the reason we get 90% accuracy on an imbalanced dataset is because we have 90% in Class-1. Most machine learning algorithms are built on statistical theory. The algorithms construct the model that says the best predictive outcome returns Class-1.

To avoid creating an imbalanced dataset, we generalize the normal operation data. For each signal bus, we randomly select 4 unique minutes of normal data such that the selected minutes do not overlap with any other signals. Within each of the four minutes, we find the moment of time associated with the maximum and minimum voltage deviation from the per-minute median (an approximation of steady state voltage) and two additional, randomly selected times from within that minute. This yields 16 samples across 54 signals, resulting in a total of 864 moments of time measured during “normal” operations. We then select 128 points in time from this repository of 864 moments for training, and an additional 128 points in time from this same repository for testing. For easily comparing the performance of machine learning algorithms on training and testing, the chosen 128 points in time are expected to yield the same amounts of examples measured by PMUs from across the power grid from each of the 128 points in time. We also ensure that there is about 20-25 (about 15-20%) points in time examined from the largest deviations. Since the algorithm does not assure to select the points of time in which all PMUs are in operation, there are 11,419
examples of normal operation collected form each of 128 points of time. An obtained point of time yields about 89 measurements on average.

4.2 Experiment objective

Before going through the evaluation of the performance of the automatically-built decision tree, it is worth exploring the characteristics of the data set. The exploration of the data set mainly relies on the observation of how its samples are collected. Recall that every feature of a sample in the data set is extracted from a fault moment of time at which a bus in the power grid experiences the largest voltage deviation. The features are then normalized with the steady state window prior to that moment of time. Since we are interested in exploring the fault signal across PMUs on a smart grid, the extraction is also processed on every PMU at the fault time. Then, for a moment of time associated with a fault event, we are sampling the features from every bus across the power grid. However, observation shows that when the power grid experiences a fault, locations far from the fault tend to show smaller voltage deviations than locations closer to the fault. Figure 4.2 illustrates the different impact on service buses across power grid system. The site furthest from a fault tends to be very similar to the state of normal operations. Thus, we expect that the decision boundary between the fault and no-fault classes will be blurred.

4.3 Developed features

4.3.1 Relative deviation features

In [8], we conduct a study to demonstrate that machine learning can be leveraged to construct a learner for line-event detection with performance that is similar to a domain experts hand-build decision rules when applied to signals located near a fault. The study also pro-
posed a set of derived expert features for the machine learning algorithm J48 to yield better performance at distance as compared to hand-built rules. The set of derived expert features is called relative deviation which are computed by using three voltages sags described in Section 2.3.2. The derived expert features represent the relative size of the second and third deviation relative to the first. Given a list of $vsag_A$, $vsag_B$, $vsag_C$, these relative features are calculated as follows:

$$vdips = \text{descendingOrder}([1 - vsag_A, 1 - vsag_B, 1 - vsag_C])$$

$$rel\_phase\_31 = \frac{vdips[3]}{vdips[1]}$$

$$rel\_phase\_21 = \frac{vdips[2]}{vdips[1]}$$

$$rel\_phase\_32 = \frac{vdips[3]}{vdips[2]}$$
4.3.2 Relative phase angle features

Under normal operation, the three voltage phases are balanced and there is a symmetrical relationship between the voltages and currents of the three phases. When the power system is imbalanced, the fault occurs and power analysis becomes more complicated. Figure 4.4 (right) illustrates a typical fault happening at line A and B. At the fault, the voltage at line C remains at normal angular displacement; but A and B have been brought closer together by the low impedance of the fault. In order to understand the fault’s altered condition, electrical engineers often transform A,B,C phase components into symmetrical components:
Positive sequence, Negative sequence and Zero sequence components [32]. The analysis of three sequence components is beyond the scope of our work in this thesis.

Most faults happening on the power grid are asymmetrical faults. They could result in producing unequal distance between the three phases. As defined in IEEE Standard and IEEE Recommended Practice for Monitoring Electric Power Quality, voltage sags have been classified into seven groups [33, 34]. Table 4.3 illustrates seven basics asymmetrical sag types in the phasor diagram. In type C, D, F and G, voltage sags affect two phase voltages and results in a change of both magnitude and angle. The relative phase angles between phases are not in 120-degree counterpart.

The set of angle changes is derived from the concept of the seven basic asymmetrical sag types above. Our angle change features are computed from the angles created by each pair of phase angles and normalized with corresponding phase angles found in the steady state window considered as the normal operation condition. Given the phase-angle-A, phase-angle-B, and phase-angle-C at the time of interest and the \( n \) phase angle PMUs measurements obtained in the steady state window, these relative features are calculated as follows:

\[
\text{ang}_{AB} = \frac{\text{relativeAngle}(\text{ang}_A, \text{ang}_B)}{\text{mean}([\text{relativeAngle}(\text{ang}_{A1}, \text{ang}_{B1}), ..., \text{relativeAngle}(\text{ang}_{A_n}, \text{ang}_{B_n})])}
\]
\[
\begin{align*}
\text{ang}_{BC} &= \frac{\text{relativeAngle}(\text{ang}_B, \text{ang}_C)}{\text{mean}(\text{relativeAngle}(\text{ang}_{B1}, \text{ang}_{C1}), ..., \text{relativeAngle}(\text{ang}_{Bn}, \text{ang}_{Cn}))} \\
\text{ang}_{AC} &= \frac{\text{relativeAngle}(\text{ang}_A, \text{ang}_C)}{\text{mean}(\text{relativeAngle}(\text{ang}_{A1}, \text{ang}_{C1}), ..., \text{relativeAngle}(\text{ang}_{An}, \text{ang}_{Cn}))}
\end{align*}
\]

\[\text{angles} = \text{sorted}([\text{ang}_{AB}, \text{ang}_{BC}, \text{ang}_{AC}])\]

### 4.4 Cascade of SVMs

The study in [8] examined the J48 learning algorithm [35] implemented by Weka [35] and trained with: 1) the set of expert derived features described in Section 2.3.2; and 2) the combined set of expert derived and relative features described in Section 4.3.1. Figure 4.5 shows the decision tree trained from (2) the combined set of features. It contains only 9 nodes, only 5 of which are leaves. The main benefit of J48 is that its classification rules can be directly compared to expert rules and validated by power system engineers. However, J48 is rarely considered the most sophisticated learning algorithm. In Section 5, we reexamine the predictive performance of this decision tree by performing error estimation on the testing dataset and detailed analysis of its false positives on a real environment.

For the rest of this section, we describe a study of employing binary-class Support Vector Machines (SVMs) [36] for our datasets. Every training example is computed, mapped into high-dimensional feature space, and separated into categories by the optimal separating hyperplane. Obtaining the best results with a SVM requires two key choices: a kernel and corresponding parameters. The subset of similarly labeled examples in the dataset forms a decision boundary. The examples lying near the decision boundary are called support vectors. The distance from the hyperplane to these support vectors on either side is called the margin, and the objective of the SVM is to maximize this margin for the best generalization.
Figure 4.5: Decision Tree Improved using Relative Deviation, a Novel Feature (sag: p.u. voltages of each phase in increasing order; dev: absolute value voltage changes from steady state measured in p.u., arranged in decreasing order.)

4.4.1 Three-stage classifier

We replicate the decision orders performed by the earlier J48 classifier into a three-stage classifier and replace its discriminant functions (single feature thresholds) with SVMs. The three-stage classifier is composed of three SVMs each of which utilizes a different subset of features for a particular type of learning. Its operation is similar to a decision tree. An unclassified example will go through three SVMs respectively: a fault-vs-no-fault classifier, a 3P-vs-other-fault classifier, and a SLG-vs-LL-fault classifier to arrive at leaf node. The cascade function contains only 7 nodes, 4 of which are leaves. It is illustrated in Figure 4.6. Next, we explore two popular kernel-based algorithms and tune them with different regularisation parameters for generating a general learner at each step.

Experiment setup

In machine learning, the goal is to generate a learner that has the highest generalization, minimal complexity and robustness that is minimally affected by external resources or variability.
Our experiment aims to construct the settings of SVM hyperparameters that demonstrate good prediction accuracy and maintain low false positive rate. We use 5-fold cross-validation on our training dataset for evaluating purposes such as choosing the better of two learning algorithms or deciding where to stop learning. Later, the testing set is used to report the estimation of expected error. It is recommended this data should not be used during development and evaluation for model selection [19]. Accordingly, for testing, we randomly leave one-fourth of the training set as the validating set. It is worth noting that because the process utilizes random factors, any individual training or testing iteration may contain exceptional instances like noise, which may lead us to draw false conclusions for optimized model settings. To generalize the learning process and lessen the impact from this, we run the process several times and report the average case performance.

We conduct the machine learning experiments on training sets and evaluate our test result on the following performance metrics described in Section 2.2.2: hamming loss, sensitivity, and specificity scores. Hamming loss intuitively informs the performance of the model in terms of accuracy; meanwhile, the sensitivity and specificity scores provide a further detailed examination when comparing models that have the same hamming loss. For example, when
choosing the setting for a fault-vs-no-fault classifier, we are interested in a learner which yields
the best response with the lowest hamming loss and the highest sensitivity score. Mostly
we discuss these scores to decide where to stop learning and finalize the hyperparameter
settings.

Figure 4.7 illustrates the process of our approach for model selection. In general, given a
list of machine learning algorithms and the training set, our experimental strategy proceeds
as follows:

1. Set a random seed and randomly split the training set into two parts: a cross-validation
   and a testing set. The testing set is one-fourth of the training set.

2. For each machine learning algorithm, we do as follows:
   • Apply 5-fold cross-validation stratification to compute the average hamming-loss
     score which is given as a validation error.
   • Train the algorithm with the cross-validation set to obtain a learner model and
test the model with the cross-validation set for training error rate.
   • Train the algorithm with the cross-validation set to obtain a learner and test the
     learner model with the testing set for testing error rate.

3. Plot and evaluate the results.

In the second step, cross-validation stratification employs a degree of randomness to
shuffle and rotate the data for statistical analysis. A random seed ensures the same set of
data rotation to make a fair comparison in the evaluation step.

Our study focuses the powerful SVM supervised learning algorithm to generate three
learners for our three-stage classifier. The key factors of an SVM’s performance are the choice
of its kernel function and/or kernel parameters. The kernel functions use these parameters
for their isolated computations to transform each data point to a higher dimensional space.
The SVM algorithm eventually generates linear classifiers in that space. The proper setting of these parameters will successfully map the input space from a non-linear system to a linear system, which could result in an easier separation of data. Another factor in performance is the penalty $C$ used by the algorithm for hyperplane formulation to control the tradeoff between margin maximization and error minimization on higher dimensional space. For large values of $C$, the optimization usually ends up getting more training data examples, which may result in more support vectors, a small-margin hyperplane and an overfitting phenomenon. Conversely, for small values of $C$, we may have underfitting.
We explore the settings for two of three popular SVM kernel algorithms implemented by popular libsvm [37]: polynomial and radial basis functions. The other linear kernel is not considered since it is a special case of radial basis kernel [38] with same performance given by some parameters \((C, \gamma)\). We drive the set of experiments by changing the most basic parameters each kernel uses. In particular, for the polynomial and radial basis kernels, we change degree and \(\gamma\) respectively; meanwhile, the penalty \(C\) remains a constant. After choosing the parameters for each kernel, we do the same experiment to study \(C\). We empirically choose SVMs for our classification problem because of the input space which can be separated by a hyperplane at the overlapped input space. Recall that a PMU measuring a nearby line event tends to produce a relatively large deviation from steady state voltages [9]. In contrast PMUs located relatively far from a fault location may measure a voltage deviation that is small enough to be nearly lost in the normal fluctuations of the system. In other words, the voltage deviation is nearly the same as the deviation of normal operation conditions.

**Fault vs No-fault SVM**

We choose a set of features \(vsags_1\) and \(angles_1\) for this classifier since they are the obvious feature set for identifying normal and line event condition. \(vsags_1\) indicates the most phase dip observed at a bus; meanwhile, \(angles_1\) represents the smallest relative angle between the two closest phases. Figure 4.8 shows the input space of these two features in which \(vsags_1\) and \(angles_1\) are shown on the x-axis and y-axis respectively. The area \((1.0, 1.0)\) is dominated mostly by normal operation examples. Several normal operations from sampling maximum voltage deviations spread out the area and share the feature space with the line event examples. During normal operation some signal buses may experience with voltage sags likely affected by line events happening far way from the managed grid. As a result, they do not get much attention to grid operators. Most line event examples are scattered along the left side and gradually merge with the normal operation area. In this case, polynomial
kernel offers better performance than RBF kernel since it can amplify the feature set to enlarge very small deviations in higher dimensional space.

Figure 4.9 depicts the experimental result from this input space. Over the cross-validation and testing set, the more complex models will give fewer errors than the relaxed models. In polynomial kernel when the degree is larger than 5.5, the model tends to show minor improvement but begins to worsen again when the degree increases above 15. The hamming loss plots show the polynomial kernel performs better than RBF kernel in terms of accuracy. In sensitivity plots, RBF kernel miss-classified about 0.05% less examples when compared to polynomial kernel. Both models indicate getting more classified fault examples could result in lower specificity core which means the more no-fault examples are miss-classified. The experimental result shows that we can choose the polynomial kernel with a degree in the range of 5.5-9.5 for this classifiers setting. In terms of data fitting, the figures show no over fitting phenomenon at any point because the cross-validation and testing curves are close together. Eventually, we chose polynomial kernel with the degree equal to 7.5 for the fault-vs-no-fault classifier.

Next, we find the C value of the classifier for our choice of kernel function parameters. We iterate over a list of values for C between 0.1 and 15 in order to learn the best margin of separation. Figure 4.10 depicts the error rate and tradeoff between the sensitivity
and specificity in this tuning process. With a larger $C$, the learning model shows a minor variance between cross-validation and training scores. The cross-validation score is likely unchanged with the larger $C$; meanwhile the training score is likely reduced. The learning model attempts to remember more training examples, which could result in overfitting. The sensitivity score has a minor increase when $C$ increases from 0.1 to 2. However, there is no inference to be made based on the score after this range because the testing examples may not be the examples near the decision boundary. The sensitivity and specificity scores are not reflected in the smaller margin space. Therefore, based on error rates, we chose $C$ equal to 1.0.

Figure 4.9: Performance metrics of feature set: $\text{vsags}_1$ and $\text{angles}_1$. The subplots at the left side demonstrate performance metrics of polynomial kernel. The subplots at the right side demonstrate performance metrics of RBF kernel.
3P vs Other Fault SVM

After classification as fault or no-fault, the fault examples will go through the 3P-vs-no3P classifier to identify whether they are a 3P fault or another type of fault. Since we are only interested in identifying fault types, normal operation examples are removed from our training set. In the same fashion, we choose a set of features, $rel\_phase_{31}$ and $rel\_phase_{32}$, for this classifier since it is the most efficient feature space that we adopted in [8]. Figure 4.11 depicts the input space of these two features in which $rel\_phase_{31}$ and $rel\_phase_{32}$ are on the x-axis and y-axis respectively. 3P examples are brought to right upper corner; meanwhile other faults are brought to the left bottom corner. The input space is linearly separable. Thus, the examples are easily separated with either polynomial or RBF kernel as shown in Figure 4.12. The performance of both kernels are similar and stable in every setting. We decided to choose RBF kernel with $\gamma$ equal to 2.0.

SLG vs LL Fault SVM

Next, we find the SVM hyperparameters for the SLG-vs-LL classifier, which is the last classifier in the cascade function. All examples are already classified except for LL and SLG faults. Similar to the previous classifier, normal operation and 3P faults are excluded in the
Figure 4.11: Feature space of sorted phase deviation after removing no-fault examples training set. Figure 4.13 shows the potential subsets of features related to the first and second sag phases. The feature spaces are distributed by two feature sets: \( \text{angles}_1, \text{rel\_phase}_{21} \) and \( \text{angles}_2, \text{rel\_phase}_{21} \) are relatively the same shape. The two fault types tend to dominate at two different poles, \((1.0, 1.0)\) and \((1.0, 0.0)\). The shape can be separated with RBF kernel. Meanwhile, the fault types rendered by other feature sets, \( \text{rel\_phase}_{32}, \text{rel\_phase}_{31} \) and \( \text{vsags}_1, \text{vsags}_2 \) tend to focus on only one pole, \((0.0, 0.0)\) or \((1.0, 1.0)\). Figures 4.14 and 4.15 depict the error rates on the two kernels for these feature sets. Due to having similar input spaces, the two feature sets \{\text{angles}_1, \text{rel\_phase}_{21}\} and \{\text{angles}_2, \text{rel\_phase}_{21}\} can be used to separate fault types with similar performance metrics. Meanwhile, the other two feature sets require more complex learning hyperparameters for performing a separation with the lowest error rates, which are still greater than the error rate generated by the previous two feature sets. Since we are looking for relaxed hyperparameters, the feature sets \{\text{angles}_1, \text{rel\_phase}_{21}\} and \{\text{angles}_2, \text{rel\_phase}_{21}\} are the two best candidates.

We choose the feature set: \( \text{angles}_2 \) and \( \text{rel\_phase}_{21} \) for the SLG-vs-LL classifier. Figure
Figure 4.12: Performance metrics of the feature set: \textit{rel\_phase}_{31} and \textit{rel\_phase}_{32}. The two subplots at the left side demonstrate performance metrics of polynomial kernel. Meanwhile, the two subplots at the right side demonstrate performance metrics of RBF kernel.

4.15 depicts the experimental results for the input space. In polynomial and RBF kernel plots, there is no gap between cross-validation and training error rates, which indicates that the learning model does not suffer from overfitting or underfitting phenomenon. The testing error rate is lower than the cross-validation and training error rate. Increasing the hyperparameters does not make any improvement. These error rates are insensitive to the increment of the hyperparameters, which may be the best error rate we may obtain for identifying SLG/LL classification. In RBF kernel, the performance metrics on both hamming loss and sensitivity are stable in terms of sensitivity and specificity. Both kernel models can be used since they offer the same performance. For the stability, we chose RBF kernel with \(\gamma\) equal to 0.5.

Next, we choose a \(C\) for our selected kernel. We perform the experiment in a similar fashion. Figure 4.16 depicts the experimental result for this process. The hamming-loss
error rate shows that our learning model does not suffer from overfitting or underfitting. According to the sensitivity and specificity rates, we chose $C$ equal to 1.0 for the larger margin space.

**Combination**

To summarize, the following setting is for our three-stage classifier:

- For fault-vs-no-fault classification, the classifier is set to employ polynomial kernel function with hyperparameter $degree$ equal to 7.5 and $C$ equal to 1.0. The classifier accepts the input space computed from the subset of features: $vsags_1$ and $angles_1$.

- For 3P-vs-other-fault classification, the classifier is set with RBF kernel function with the hyperparameter $\gamma$ equal to 2.0 and $C$ equal to 1.0. The subset of features used for this classifier is $rel\_phase_{31}$ and $rel\_phase_{21}$.

- For SLG-vs-LL classification, We set its hyperparameters: RBF kernel function and $\gamma$
equal to 0.5 and $C$ equal to 1.0. The classifier uses the input space computed from the subset of features: $\text{angles}_2$ and $\text{relative\_phase}_{21}$.

Building the three-stage classifier relies on the observation that every fault type can be represented by using a different particular subset of features. As a result, we can use different hyper-parameters for generating a better classifier. The upper classifier in the three-stage classifier tests an instance to obtain a type of classification and passes the instance to a further classifier if its outcome prediction is negative. These classifiers are independent in learning; but lower classifiers strongly rely on the outcome of the upper classifier to filter out some data. For instance, if an instance is predicted as a fault, the 3P-vs-other-fault classifier will learn the variables from the subset of features $\text{rel\_phase}_{31}$ and $\text{rel\_phase}_{21}$ and ignores the variable $\text{vsags}_1$. 3P-vs-no-fault relies on the fault-vs-no-fault classifier properly filtering out no-fault data, since the subset of features used in the 3P-vs-no-fault classifier is only capable of reliably separating 3P faults from other faults, not from normal operation data.

4.4.2 Five-stage classifier

Our fault-vs-no-fault classifier computes its model using the following subset of features: $\text{vsags}_1$ and $\text{angles}_1$ over the entire training dataset. The model implicitly recognizes all fault examples having the same feature set distribution. In contrast, our observation shows that our training dataset has unequal distribution of $\text{vsags}_1$ at the range that is close to 1.0 for the three types of faults. The histograms in Figure 4.17 illustrate the frequency of fault examples found at the range between 0.98 and 1.0. There are about 7 LL examples appearing at the range 0.995 to 1.0; meanwhile, there are no 3P examples appearing in the same range. Since SVMs learning algorithms are instance-based, decision boundaries are formed by the distribution of instances over their feature space.

From our observation, it is worth exploring a five-stage classifier as shown in 4.18. We
decompose the fault-vs-no-fault problem into different fault/no-fault classifiers for the “all-together” method. Our method is also called “one-against-one” learning strategy which offers a more appropriate and practical approach [30]. Deciding fault-vs-no-fault classification last, rather than first, may generate fewer expected false positives in the positive classification since different fault/no-fault decision boundaries could be found once the primary characteristics have a chance to group the examples into their most likely constituent type.

In Section 5, we evaluate the performance of this cascade function and compare it with our three-stage classifier. The experiment proves that the five-stage classifier offers superior performance in eliminating false positives in real-data detection while maintaining relatively high true positive detection performance.

For the purpose of comparison, the hyperparameters of the SVMs in our model are kept the same as the three-stage classifier such that the fault/no-fault classifiers are set to employ polynomial kernel with a hyperparameter $degree$ equal to 7.5.
Figure 4.14: The error rates on two kernels: polynomial and RBF by using the sets of features (a) $\text{angles}_1$ and $\text{rel\_phase}_21$, (b) $\text{rel\_phase}_31$ and $\text{rel\_phase}_32$, and (c) $\text{vsags}_1$ and $\text{vsags}_2$
Figure 4.15: Performance metrics of feature set: $\text{angles}_2$ and $\text{rel\_phase}_{21}$. The subplots at left sides demonstrate performance metrics of polynomial kernel. The subplots at right sides demonstrate performance metrics of RBF kernel.

Figure 4.16: Performance metrics of feature set: $\text{angles}_2$ and $\text{rel\_phase}_{21}$ with the RBF kernel. The left subplot illustrates the change of the error rate when the larger $C$ is set, and the right subplot shows the sensitive and specificity in the context of tuning process.
Figure 4.17: Unequal distribution of $vsags_1$ in the range between 0.98 and 1.0. The distribution of $vsags_1$ in different fault types SLG, LL and 3P are showed respectively.

Figure 4.18: Five-stage classifier
Chapter 5

Evaluation

In this section, we report error rate estimation and somewhat different tradeoffs in terms of accuracy and true positive/false positive rate. The study of error rate estimation previously studied J48 decision tree (Section 4.4), proposed multiple-stage classifiers (Section 4.4 and 4.4.2) and a standard “one-against-one” SVMs approach for multi-class learning on the testing dataset generated one-time in Section 4.1. The “one-against-one” method is the default approach for learning multi-class problems with Support Vector Machines in Python’s scikit-learn version 0.16 [39], which we leverage for our work. In our domain, there are four classes (three fault classes and the normal operation class), and the “one-against-one” method learns six binary classifiers covering the \( \binom{n}{2} \) class pairs. The classifiers then vote on the final prediction. In this section we also explore several supervised learning algorithms and report their performance for our approach. Next, we examine the multiple-stage classifiers on one day of data from normal operation which has not been not examined before.
5.1 Examining the error rates

We measure the error rates for a J48, standard “one-against-one” SVMs, three-stage SVM classifier and five-stage SVM classifier on our testing dataset. The training dataset described in Chapter 4 is used to train the multiple-stage classifiers. Three trained classifiers perform outcome prediction on our testing dataset. We choose polynomial kernel with degree equal to 7.5 for the standard SVMs algorithm. The subset of features for this classifier includes $vsags_1$, $angles_1$, $rel\_phase_{31}$ and $rel\_phase_{32}$. These features are shown to be very useful for identifying fault types.

Confusion matrices for predictive performance of the classifiers are reported in Table 5.1. The performance of our four proposed classifiers shows relative similarity in terms of accuracy; among these, the J48 classifier offers superior performance with 95.5%. In contrast, the five-stage classifier reduces false positives to 1.7%, nearly a factor of 3 compared to the other classifiers. The J48, standard SVMs and three-stage classifier produce a false positive rate of 4.9%, 4.5% and 4.5% respectively. Accordingly, the five-stage classifier offers a better trade-off in terms of false positive rate compared to all other examined classifiers.

From this experiment, we have learned:

1. The structure of J48 which only uses one threshold value for discriminating all fault types and no-fault classification causes more errors in separating normal operation with LL and 3P faults than the rest classifiers. Both J48 and three-stage classifiers share the same learning configuration. The fault/no-fault classification is composed of one determinant function, which results in highly accurate classification of 3P examples in the testing dataset. On the other hand, they also show a false-positive rate of about 4.0% and 3.8%.

2. The “one-against-one” classifier using the combined feature set is not ideal for this problem. The predictive outcome of this approach misses more the LL and 3P faults
Table 5.1: Testing dataset performance of the proposed classifiers

than other methods. Although this model’s identification rate of true SLG faults is highest, the number of false positives is also high, which could result in making the model impractical in real detection.

3. J48 has excellent performance at identifying SLG examples. Recall that J48 identifies fault/no-fault classification with a threshold value. The predictive fault examples go into further test functions for classifying other types of faults (Figure 4.5). There are only 96 no-fault examples that pass this threshold value to become a false positive at the SLG leaf. Also, only 146 fault examples end up as false negatives in the no-fault leaf. Although our multiple-stage classifiers offer a relatively similar false positive count at the SLG leaf, they also predict significantly more false negatives.

Our five-stage classifier performs the best in terms of minimizing the false positive rate.
<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLG</td>
<td>6914</td>
</tr>
<tr>
<td>LL</td>
<td>94</td>
</tr>
<tr>
<td>3P</td>
<td>1</td>
</tr>
<tr>
<td>NF</td>
<td>93</td>
</tr>
</tbody>
</table>

Accuracy score: 96.61%
Fault/no-fault false positive: 1.8%

Table 5.2: Testing dataset performance of new 5-stage proposed classifiers

But the classifier misses SLG examples the most when compared to others. Deriving from this observation, we copy J48 structure at SLG classification for our five-stage classifier. Figure 5.1 illustrates the structure of new five-stage classifier. Like the process of five-stage classifier, a predicted example normally goes through each stage of classification and returns the classification result at the leaf node. If the example is predicted as a possible SLG fault, the classifier compares the example’s $vsags_1$ with its threshold value to determine the SLG/nofault classification. We use 0.9981 threshold value for identifying fault/nofault classification. The value is obtained from the J48 algorithm. As the result, the performance described in Table 5.2 to classify SLG is enhanced significantly. There are 193 SLG examples to be predicted as nofault examples instead of 778 examples as performed by five-stage classifier.

5.2 Performance of alternate classifiers

5.2.1 Motivation

We want the most accurate learner with our dataset. The learner is generated by machine learning algorithms which may come up with different mathematical models. As stated by the No Free Lunch Theorem [40], there is no best algorithm [41]. For any learning algorithm,
Figure 5.1: Improved-five-stage Cascade of SVMs

there is a dataset where it performs well and another dataset where it performs poorly. Caruana et. al propose the performance study to rank the machine learning algorithms: boosted tree, random forest, bagged trees, SVMs and neural nets respectively [41]. The study is derived from the well-known project STATLOG [42]. The purpose of the study provides machine learning practitioners a list of efficient learning algorithms which are able to yield the best result in general problems. The experiments explore the performance of these algorithms and evaluate the performance with different metrics on large-scale real-world problems. The performance is evaluated on a broad set of performance metrics to adapt the diversity of business domains. In this section, we take these algorithms into account to study primarily the input space and evaluate different learning models with respect to our learning objective.

5.2.2 Experiment setup

The experiment should have a good mix of algorithm representations. We use the same training and testing dataset described in Section 4.1 for this experiment. Also, the subset of features for examined alternative classifiers includes $vsags_1$, $angles_1$, $rel\_phase_{31}$ and
rel_phase\textsubscript{32} since these features are shown to be very useful for identifying fault types. We choose the following algorithms available in Weka tool [29]:

- Linear methods: Logistic Regression (LR).
- Non-linear methods: Naive Bayes (NB) and NB with kernel estimator (NB-E).
- Non-linear decision tree and Rule: SimpleCART and PART.
- Ensembles of trees: Random Forest (RF), Regression Tree (RT).

We also report the performance of non-linear decision trees with different ensemble methods: Bagging (B) and AdaBoostM1 (AB) according to the suggestion of [41]. Our objective evaluation is Accuracy, True Positive Rate and False Positive Rate on Boolean fault/no-fault classification in which fault and no-fault classifications represent positive and negative class respectively. To compare with our existing classifiers, these candidate classifiers are trained as a multi-class problem. We later convert their results into binary fault/no-fault classification result and compute the performance metric scores.

We choose the best parameter settings for the decision tree algorithms using cross-validation on our training dataset; then we report the metric scores on the final testing dataset. NB and NB-E do not have hyperparameters. The LR’s hyperparameter is set to default.

SimpleCART, PART and RT are pruned during cross-validation to select the best tree for each. We want to generate a tree that can give out suitable predictions. Having a high accuracy decision tree for training data results in a larger tree. We do not expect to have a highly reliable tree in this case because these thresholds will make the tree too strictly defined and therefore the tree will have difficulty properly categorizing real-life events. To have a high reliability tree, the algorithm will try to absorb as many different characteristics of events in the training dataset as possible. This kind of training is remembering. This is
<table>
<thead>
<tr>
<th>Model</th>
<th>Ensemble method</th>
<th>Accuracy</th>
<th>True positive rate</th>
<th>False positive rate</th>
<th>Tree size</th>
</tr>
</thead>
<tbody>
<tr>
<td>J48</td>
<td>.963</td>
<td>.981</td>
<td>.049</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>.954</td>
<td>.952</td>
<td>.045</td>
<td>7*</td>
<td></td>
</tr>
<tr>
<td>three-stage</td>
<td>.936</td>
<td>.907</td>
<td>.045</td>
<td>7*</td>
<td></td>
</tr>
<tr>
<td>five-stage</td>
<td>.930</td>
<td>.939</td>
<td>.017*</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>improved-five-stage</td>
<td>.966</td>
<td>.941</td>
<td>.018*</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>NB</td>
<td>.953</td>
<td>.952</td>
<td>.045</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NB-E</td>
<td>.973*</td>
<td>.977</td>
<td>.029</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LR</td>
<td>.958</td>
<td>.957</td>
<td>.041</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SimpleCART</td>
<td>.973*</td>
<td>.987</td>
<td>.036</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>SimpleCART B</td>
<td>.977*</td>
<td>.994</td>
<td>.033</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>SimpleCART AB</td>
<td>.972*</td>
<td>.976</td>
<td>.030</td>
<td>147</td>
<td></td>
</tr>
<tr>
<td>PART</td>
<td>.977</td>
<td>.986</td>
<td>.028</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>PART B</td>
<td>.979*</td>
<td>.988</td>
<td>.028</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>PART AB</td>
<td>.974*</td>
<td>.979</td>
<td>.028</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>RF</td>
<td>.971*</td>
<td>.982</td>
<td>.035</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF B</td>
<td>.975*</td>
<td>.983</td>
<td>.031</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF AB</td>
<td>.976*</td>
<td>.986</td>
<td>.030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RT</td>
<td>.973*</td>
<td>.996*</td>
<td>.043</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>RT B</td>
<td>.975*</td>
<td>.995*</td>
<td>.039</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>RT AB</td>
<td>.976*</td>
<td>.983*</td>
<td>.03</td>
<td>67</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3: Testing dataset performance of studied classifiers. The boldface marker at a score indicates the best score. The * marker at a score indicates the score closest to the best.
not an inappropriate thing to do because the collected dataset is finite. Pruning the tree effectively leaves out the details and retains the most common differences in the tree. This increases the minimum number of items per leaf. We try to learn the common characteristics of events, but we do not attempt to keep as many differences as possible.

For RFs, we examine the “number of trees” setting by starting it with 2 and doubling it. The empirical examination shows that accuracy increases significantly as the number of trees is increased up to six, after which the accuracy gained by adding trees is minor. Therefore, we choose 6 for the number of trees for RF.

5.2.3 Results

We should note that line events are relatively infrequent, at least on a transmission grid such as the Bonneville Power Administration’s. Line events occur on the order of approximately one hundred per month distributed across their grid. Given that each PMU generates over 5 million samples per day, even a single PMU could easily overwhelm a human operator with false alarms if the false positive rate on the classifier is not kept sufficiently low. Indeed, even a false positive rate of 0.017 (as measured by the five stage cascade in Figure 5.1) could lead to 85,000 false positives on a single day. Thus, the models with a reduced false positive rate are worth further investigation.

Table 5.3 shows the scores for each algorithm on each of the three metrics. The algorithm with the best performance on each metric is boldfaced. Other algorithms whose performance is close to the best one is marked with *. In general, an algorithm with the ensemble methods do not show any improvement over the original. In the worst case, AB increases the tree size for “remembering” the examples. Most examined algorithms are shown to perform better than our proposed classifiers in terms of accuracy. PART is considered the best in the list since it produces the least false positive rate and highest accuracy. However, it
presents relatively high in false positive rate (at least double) when compared to five-stage and improved-five-stage classifiers. The score predicts that PART may not be the classifier we are looking for.

It is unknown which classifier is the best for our approach. In Section 5.4, we study the false positive rates of J48, five-stage, and improved-five-stage classifiers on real detection. We are interested in seeing these rates on 1-day normal operations. The five-stage classifier provides us the lower bound of misclassification on a normal day. Meanwhile, the others provide the middle- and upper- bounds. Improved-five-stage classifier is worth exploring since it has excellent performance. The result is used to infer the best classifier for our approach.

5.3 Towards full-time operations

5.3.1 Motivation

To further minimize false positives, we explore two modifications to the original training dataset: over-sampling and under-sampling methodologies. Each of the methodologies has received significant attention to counter the effect while generating the learner. In particular, the over-sampling methodology boosts the minority class distribution which can lead to overfitting on the multiple copies. Among the most well used is SMOTE [43] which performs synthetic oversampling of the minority class. However, the problem faced in this domain is not so much a lack of minority class data as the sheer scale of the imbalance. Additionally, our proposed multiple-stage classifiers to learn the minority LL and 3P classes are efficient. In 3P classification given in 5.1, the three-stage classifier gets 259/267 correct hits and J48 classifier retrieves 266/267 hits. These experimental results show that our imbalanced data examples do not deter us from learning minority classes. It is not necessary to explore this
method for our study. Meanwhile, the over-sampling methodology typically employs a degree of randomness to remove some examples which can be significantly important data examples. Instead of randomly selecting examples to drop, understanding our data collection gives us a deterministic choice. We discuss this method below.

To further minimize false positives, we explore an under-sampling method to the original training dataset. One approach would be simply to ignore measurements from PMUs that are geographically distant from a fault location. Intuitively, this should not be problematic from a detection standpoint since our dataset contains a large number of PMUs over a wide geographical area (covering two states in the Pacific Northwest of the United States of America). Removing a portion of the PMU measurements should still ensure that many measurements are still available for classifying the event. Intuitively, removing fault examples according to this idea could also result in reducing true positives since we lose the information of measurements far away from a fault location.

5.3.2 Experiment setup

In this case study, we investigate how the false positive rate is impacted by modifications to the training data. We examine our five classifiers: J48, one-vs-one standard SVMs, three-stage, five-stage and improved-five-stage classifiers. We respectively drop only fault examples in the training dataset by thresholding $vsags_1$ in the range for all iterations. At every iteration we retrain our classifiers with the dataset after removing fault examples. Every trained model is tested on an unmodified testing dataset; then we report its testing score on a resulting line in a ROC style plot. Since we are interested in true/false positives on the boolean fault/no-fault classification, we compute the performance metrics as described in the case study 5.2. There is a special case for J48 classifier whose decision nodes are described in Figure 4.5. We only change the threshold on the first decision node which
corresponds to the fault/no-fault distinction; we do not retrain the classifier as a whole.

Thresholding \( vsags_1 \) with the range of 0.95 and 0.999 would remove some significant sagged signals which represent the line events. We choose 0.95 for the lower bound based on our expert decision rule in Figure 2.3. The upper bound of the range is the upper bound of \( vsags_1 \) empirically observing SLG examples.

### 5.3.3 Results

Figure 5.2 shows the effect of removing measurements whose voltage deviation were below some threshold from the training set. Specifically, the figure plots a representative point for each of the five classifier’s performance as represented in the confusion matrices in Tables 5.1 and 5.2. In general, five-stage and improved-five-stage classifiers perform best overall, showing the lowest false positive rate with the full training dataset (upward facing triangular and “+” points). Improved-five-stage classifier offers the most superior performance in terms of highest true positives while keeping false positives relatively similar to the five-stage classifier with full training set. The plot also reveals that the one-vs-one SVM tends to outperform the 3-Stage cascade for false positive rates between roughly 0.8% and 4.5% respectively. However, like J48 and the 3-Stage classifier, it does not quite yield the same true-positive performance as the 5-Stage classifiers.

It is clear that dropping fault examples at a distance also results in significantly reducing the fault detections of classifiers. For the best classifiers, the false positive rate was roughly reduced from 1.8% to nearly 0%, but the true positive miss rate was increased to more than 40%. Because of this loss of accuracy, it is obvious why we choose not to further explore the method of changing training data. Recall that our testing datasets are collected partly from measurements at the moment of time with the largest voltage deviations during 800 normal operation minutes. We compress this data into more than 11,000 examples. Because
of the selection process, these examples may not truly represent normal operations in a real environment. The false positives may be insufficient to represent real-world performance in which the measurements are generated at a large-scale.

![ROC curve for proposed classifiers](image)

Figure 5.2: ROC curve for proposed classifiers

Our goal is to detect line events on the smart grid PMU data stream. Most previous experiments try to classify the fault signals in the grid, and report the classification results as individual events in the grid. For signals measured by PMUs that are far from the fault location, the voltage signature tends to be similar to the signature of normal operations. In this experiment, we are interested in a strategy for reducing the false positive rate by corroborating fault classifications to constitute a final decision. This strategy may give a higher degree of confidence to determine if line events have occurred by surveying a number of classifications from multiple signals. We corroborate these fault classifications at the specific moment of time to make this determination. Accordingly, a set of examples which are sampled at a moment of time $t$ are classified using our proposed classifiers. These resulting classifications are then used to infer the final classification of time $t$. The easiest method of
constituting the final result is based on the proportion of signals experiencing the fault. In other words, the system decides that a fault has occurred if the number of signals classified as fault is larger than a threshold. The constituted fault classification rule is illustrated in the following equation in which $v$ represents the threshold value of corroborated fault classification.

$$\sum_{i} \text{class}[t][i] > v$$

Figure 5.3: The tradeoff between true positive rate and corroborated fault classification. The false positive rates are annotated at the points in which the rates are changed as the threshold value is increased on the y-axis.

Figure 5.3 illustrates the experimental results for the threshold values in a range of $(0.05, 0.8)$. The proposed classifiers are trained with unmodified training dataset. Given a trained classifier and threshold value, we perform fault/no-fault classification on the set of examples at a time $t$ in the testing dataset, calculate a summation of fault classifications, and test the summation against the threshold value. We do the same process for every moment.
of time found in the testing dataset. The truth classification at \( t \) is the real classification of the event at that time. We plot the true positive rates and the threshold values on the x-axis and y-axis respectively; meanwhile, the false positive rates are annotated accordingly. We only annotate the false positive rates where the rate changes as we increase the threshold values. When increasing the threshold value, we expect the false positive rate will decrease accordingly, since a larger threshold value will require more fault signals to classify that time as a fault.

The corroborated fault classification on J48 is quite insensitive to the change of the threshold variable. This is because its original classification result produces a very high true positive rate. Both J48 and three-stage classifiers yield relatively high false positive rates (0.47 and 0.39 respectively). Meanwhile, the five-stage and improved five-stage classifiers produce very low false positive rates. The five-stage classifier yields the lowest true positive rate (0.75) when the threshold variable is set to 0.8. The improved five-stage classifier appears to be a good candidate in this experiment since it offers a relatively high true positive rate compared to the five-stage classifier with the same threshold.

5.4 Normal operation test set

5.4.1 Motivation

The ROC curve in Figure 5.2 shows that no classifier outperforms the five-stage classifiers. In this section, we explore the five-stage SVM classifiers’ performance as we scale the test data to simulate a real operating environment.

Our previous case studies examine only a small sample of normal operating data. As a result, we expect that the estimate of our observed false positives may be incorrect by a significant margin. Thus, in this experiment we are interested in a contiguous 24 hour
period in which there is no line event recorded. We want to examine the performance of the five-stage classifiers to determine how the false positive rate compares to our much smaller test set previously examined. The classifiers are trained by our unmodified training dataset. These trained classifiers produce the least false positives in the range of 1.7% and 1.8%. To provide a base line, we also perform the same large scale test with the original J48 classifier which produces a 4.9% false positive rate. These chosen classifiers also give us the low, middle and upper bound false alarm with the understanding that abundant false alarms will limit the value of any classifier outside of the laboratory.

5.4.2 Experiment setup

We use signals from 19 PMUs across the grid over a contiguous 24 hour period. Each PMU reports 5,184,000 measurements over a 24 hour period, however one PMU is offline for 283 cycles (4.7 seconds) during the period we selected. Additionally, we use the first 10-cycles from each PMU to acquire a steady state value for each voltage measurement. In total this means that there are (5.184 · 10^6 · 19) − 283 − (10 · 19) ≈ 98.5 · 10^6 individual examples to classify. However, from the analyst’s standpoint, it is the moments of time that matter, not the individual measurements, so we consider only the unique moments where all PMUs report valid signals (i.e., 5.184 · 10^6 − 293 moments).

Our classifiers can not utilize this test set directly to perform a prediction. The test set needs to be transposed into data points, each of which is comprised of the feature values evaluated by our classifiers. Most of our domain features to be computed require a steady state window which explicitly contains the measurements with no-fault condition. It is unwise to localize a steady state window for every measurement. The computation is expensive and inefficient for real-time detection. We design the feed forward algorithm to move the steady state window along with the prediction. Within this section we also explore three schemes
for governing this movement. We discuss these in turn below.

- Standard steady state window movement: The obvious approach is to leverage the predictive outcome at time $t - 1$ for time $t$ to determine if the steady state window can be moved. Accordingly, the progress of finding faults is demonstrated as follows. Given a signal, we begin with establishing a steady state for each phase found in the signal window. The algorithm of finding the steady state window is mentioned in Algorithm 1. For each the moment of time right after the end of the found steady state window, we compute the feature set utilized by the classifiers and perform classification using the computed values of the feature set for a classifier. During the progress, we save the moments of time at which the classifier reports the fault condition and its type. After prediction, we update the steady state window for the next predictive performance by appending the current measurement and removing the first measurement from the steady state window. Updating is only be permitted if the predictive outcome is no-fault. If a fault is predicted, the window remains stationary until normal operation resumes. The same technique is applied to each minute and for each classifier.

- Sophisticated steady state window movement: This scheme uses the same basic method as the standard scheme, but reports “steady state error” if the steady state window lags more than three measurements behind the current point. This means the algorithm will defer its fault detection and recompute a new window steady state from the current point. The scheme is used to counter an adverse affect of the standard window moment illustrated in Figure 5.4. If the voltage sags at triple phases visualized by the vertical line near cycle 2,700 caused a contiguous string of false positives, this could result in the steady state measurement lagging significantly far back in time (and thus unlikely to be precise enough for the decision tree). Our goal for this scheme is to ensure that such lagged measurements don’t trigger false positive while also being conservative so
that the steady state window does not inadvertently incorporate part of a fault.

- Continuous steady state window movement: The scheme simply uses the previous 30 measurements for the steady state window, regardless of the smoothness of the measurements during that window, and regardless of the classification within that window. Since the window always keeps pace with the measurement, we expect that transient noise may create some false positives, but eventually the signal will smooth out and classification will continue regularly. We increased the window length from 10 to 30 cycles to avoid being overly influenced by isolated measurement outliers.

![Figure 5.4: A steady state problem during classifying.](image)

5.4.3 Results

With the first scheme of moving steady state window, five-stage classifier produces 15 false positive moments of time, but improved-five-stage and J46 classifiers produce 4574 and 4,582,490 false positive moments of time. Our initial hypothesis is that one PMU may have been behaving erratically during the day we selected. To test this, we record the number of PMUs whose signals are classified as “fault condition” for each moment in time. We then apply a floating threshold so as to require multiple PMUs to corroborate each others’ classification before marking a particular moment in time as a fault. We test thresholds of
1, 2, 3, and > 3 PMUs. In the first case, we ask how many times exactly 1 PMU classifies a moment as a fault; in the last case, we ask how many times there are more than 3 PMUs classifying the same moment as a fault. These results are shown in the first three rows of Table 5.4. The first column shows how many moments cannot be classified (because they are used to initialize the steady state measurement or because a PMU is out of service); while the second column shows how many moments are classified as “normal” by all PMUs.

Since our selected signals are based on the topological distance matrix [9] such that no pair of any sites are adjacent, the stratification tells us the affected level of false alarms propagating on the entire smart grid and the tradeoff in terms of reducing false positives by using extra information from the PMU corroboration data. At any moment of time, if there is only one PMU reporting false, the system tends to experience a local false alarm which occurs at one or two sites. If our final predictive decision requires more corroboration, the next columns inform us an estimate of the error rate that each classifier is likely to give.

The results across all three steady state schemes indicate that the five-stage classifier is relatively insensitive to the method of steady state calculation, while J48 performance is much more sensitive, even as the amount of corroboration required between PMUs is increased. Overall, the results from the five-stage classifier are quite promising in this streaming environment. Across the 24 hour period, there are only 2 moments in time (regardless of the steady state method employed) that are incorrectly classified as faults by more than three PMUs when using the five-stage classifier. This false positive rate is 0.0000386% — more than four orders of magnitude less than expected given the results from the original test set.

Although the performance of both of the five-stage classifiers on our initial testing dataset is relatively similar, the improved-five-stage classifier is more sensitive to the steady state schemes than the five-stage classifier at any point of corroboration requirement. The experiment demonstrates that our five-stage classifier provides a very clean decision boundary for learning the fault/no-fault concept. Improving the five-stage classifier to get more true
<table>
<thead>
<tr>
<th></th>
<th>Out of Service</th>
<th>Report Fault By</th>
<th>0 PMU</th>
<th>1 PMU</th>
<th>2 PMUs</th>
<th>3PMUs</th>
<th>&gt; 3PMUs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Five-stage classifier</td>
<td>293</td>
<td>5,183,692</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0057%</td>
<td>99.991%</td>
<td>0.0002%</td>
<td>0.0001%</td>
<td>0.0000%</td>
<td>0.0000%</td>
<td></td>
</tr>
<tr>
<td>Improved-five-stage classifier</td>
<td>293</td>
<td>5,179,133</td>
<td>4546</td>
<td>20</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0057%</td>
<td>99.9061%</td>
<td>0.0877%</td>
<td>0.0004%</td>
<td>0.0001%</td>
<td>0.0001%</td>
<td></td>
</tr>
<tr>
<td>J48 classifier</td>
<td>293</td>
<td>601,217</td>
<td>139,196</td>
<td>226,854</td>
<td>285,764</td>
<td>3,930,676</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0057%</td>
<td>11.5976%</td>
<td>2.6851%</td>
<td>4.3760%</td>
<td>5.5124%</td>
<td>75.8232%</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sophisticated SS window movement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Five-stage classifier</td>
<td>333</td>
<td>5,183,653</td>
<td>10</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0064%</td>
<td>99.9933%</td>
<td>0.0002%</td>
<td>0.0000%</td>
<td>0.0000%</td>
<td>0.0000%</td>
<td></td>
</tr>
<tr>
<td>Improved-five-stage classifier</td>
<td>1,094</td>
<td>5,182,548</td>
<td>338</td>
<td>13</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0211%</td>
<td>99.9719%</td>
<td>0.0065%</td>
<td>0.0002%</td>
<td>0.0001%</td>
<td>0.0001%</td>
<td></td>
</tr>
<tr>
<td>J48 classifier</td>
<td>1,763</td>
<td>5,181,655</td>
<td>527</td>
<td>26</td>
<td>11</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0340%</td>
<td>99.9548%</td>
<td>0.0102%</td>
<td>0.0005%</td>
<td>0.0002%</td>
<td>0.0003%</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous steady state window movement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Five-stage classifier</td>
<td>313</td>
<td>5,183,669</td>
<td>13</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0060%</td>
<td>99.9936%</td>
<td>0.0003%</td>
<td>0.0001%</td>
<td>0.0000%</td>
<td>0.0000%</td>
<td></td>
</tr>
<tr>
<td>Improved-five-stage classifier</td>
<td>313</td>
<td>5,182,883</td>
<td>777</td>
<td>20</td>
<td>1</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0060%</td>
<td>99.9785%</td>
<td>0.0150%</td>
<td>0.0004%</td>
<td>0.0000%</td>
<td>0.0001%</td>
<td></td>
</tr>
<tr>
<td>J48 classifier</td>
<td>313</td>
<td>5,181,968</td>
<td>1,440</td>
<td>130</td>
<td>41</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0060%</td>
<td>99.9608%</td>
<td>0.0278%</td>
<td>0.0025%</td>
<td>0.0008%</td>
<td>0.0021%</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4: False positive found by corroborated classifications

Positives could result in simultaneous increase in false positives. We also have shown that the five-stage classifier achieves a very low false positive rate on a 24 hour period of data containing almost 100 million examples. We conclude from our experiments that the five-stage classifier is the best classifier for our approach.
Chapter 6

Conclusion and Future Work

6.1 Conclusion

The development of smart grid technology PMUs presents new opportunities to monitor the state of the power system across wide geographical areas in real-time. This technology also presents a challenge to work with its large-scale generated data. In the thesis, we leveraged the Bonneville Power Administration’s (BPA’s) archive of large-scale historical data to introduce an approach for constructing an efficient machine learning event detector which uses its current PMU installation.

Our works derive from an empirical study by our domain expert for a rule-based wide-area fault detection method and our publication to demonstrate that machine learning algorithms can be used to leverage much more data to perform a better line event detection than the hand-built rules. The learner generated in our earlier work is used to provide the baseline performance for the results in this thesis. By leveraging archival BPA’s synchrophasor data and event signatures associated with this data, we construct training and testing datasets for our event detectors and propose three multiple-stage cascade classifiers for our problem by employing machine learning techniques and Support Vector Machine algorithms. The
efficiency of these learning models is respectively evaluated in turn as follows: (i) studying the estimate of error rate of these models using the separated testing dataset collected in same synchrophasor data; (ii) examining and comparing the performance of other popular machine learning algorithms and ensemble methods for our problem; (iii) performing an under-sampling method to study the impact of modifications to our training dataset; and (iv) performing a detailed analysis of false alarms and exploring multiple methods in steady state calculation for reducing false alarms in a real system.

Through these studies, we demonstrate that our five-stage cascade classifier offers a better performance profile than reported classifiers for the same domain. The model can be used for real event detection since it can perform a wide-area fault/no-fault detection with high accuracy (93% in testing dataset) and very low false positive rate (0.0000386% in new 1-day test set). Thus, the five-stage cascade classifier is quite promising in this streaming power grid domain.

### 6.2 Future Work

The beauty of our classifier is that every stage can be changed with different types of machine learning algorithms, which could improve overall performance. There are three SVM classifiers comprised for identifying fault/no-fault classification. Since the classifiers are configured with the same hyperparameters and subset of features, they could be replaced with one-class SVM classifier for detecting anomaly or outliers. In this case one-SVM classifier takes input space of normal operation for its training phrase and predicts any fault types. Implemented according to this approach, the five-stage classifier may reduce its cascade structure and computation a complexity.

While the prototype demonstrated in this thesis has relied on the voltage sag input at a moment of time to perform predictions, some other data mining techniques can be exploited
to leverage much more synchrophasor data by using signatures that are computed a window of time.
Bibliography


