A Comprehensive Evaluation of Supervised Machine Learning for the Phase Identification Problem

Brandon Foggo, *Student Member, IEEE* and Nanpeng Yu, *Senior Member, IEEE*

***Abstract*—Power distribution circuits undergo frequent network topology changes that are often left undocumented. As a result, the documentation of a circuit's connectivity becomes inaccurate with time. The lack of reliable circuit connectivity information is one of the biggest obstacles to model, monitor, and control modern distribution systems. To enhance the reliability and efficiency of electric power distribution systems, the circuit's connectivity information must be updated periodically. This paper focuses on one critical component of a distribution circuit's topology - the secondary transformer to phase association. This topology component describes the set of phase lines that feed power to a given secondary transformer (and therefore a given group of power consumers). Finding the documentation of this component is call Phase Identification, and is typically performed with physical measurements. These measurements can take time lengths on the order of several months, but with supervised learning, the time length can be reduced significantly. This paper compares several such methods applied to Phase Identification for a large range of real distribution circuits, describes a method of training data selection, describes preprocessing steps unique to the Phase Identification problem, and ultimately describes a method which obtains high accuracy ($> 96\%$ in most cases, $> 92\%$ in the worst case) using only 5% of the measurements typically used for Phase Identification.**

***Index Terms*—Distribution Network, Machine Learning, Network topology, Phase Identification, Smart Grid**

NOMENCLATURE

$\mathbf{a}$	Neural nonlinearity input.
c	Class.
$\mathcal{C}$	Class set.
$\mathcal{D}$	Dataset.
$\mathcal{D}^*$	Dual dataset.
d	Dimensionality.
f^*	Perfect predictor function.
$\mathcal{H}$	Hypothesis space.
J	Objective function.
$\mathcal{K}$	Set of K-nearest neighbors.
L	Likelihood function.
m	Feature.
$\mathcal{L}$	Index set of labeled data.
p	Probability density / mass function.
q	Variational distribution.
S	Representative set of power consumers.
S^*	Representative set of features.
t	Target.
$\mathcal{U}$	Index set of unlabeled data.
W	Neural Network transformation matrix.
$\mathbf{w}$	Evidence vector.
X	Design matrix.
$\mathcal{X}$	Input space.
$\mathbf{x}$	Voltage time series.
$\tilde{\mathbf{x}}$	Voltage time series with pre-appended 1.
y	Output class.
$\mathbf{z}$	Neural nonlinearity output.
δ	Discrete Kronecker delta function.
θ	Parameter set.
λ	Bandwidth of Radial Basis Function Kernel.
μ	Mean.
Σ	Covariance matrix.
σ	Nonlinear activation function.
$\#$	Cardinality.

This work was supported in part by the National Science Foundation (NSF) under Grant 1637258 and in part by Department of Energy (DOE) under Award DE-EE0007328.

B. Foggo is with the University of California, Riverside (email: bfogg001@ucr.edu)

N. Yu is with University of California, Riverside

I. INTRODUCTION

Power distribution circuits undergo frequent topology changes that are often left undocumented. As time passes, all current documentation of the circuit's topology become unreliable. But this topology documentation is critical to the operation and planning of power distribution circuits. For example, power flow analysis, state estimation, and Volt-VAR control all depend on accurate topology information. For this reason, the documentation is typically updated by periodic field testing projects scheduled by the circuit's utility company. However, these tests can take time lengths on the order of months. As a result, there are significant periods of time in which the documentation of a distribution circuit is incorrect. A faster and less expensive method of estimating distribution circuit topology is necessary.

The topology of a distribution circuit is organized into three hierarchical levels. At the *primary* level, primary feeders partition the distribution circuit into relatively large sets. At the *lateral* level, each primary feeder connects to a set of step down transformers which are connected through some combination of the feeder's three phases. Finally, at the *secondary* level, each step down transformer connects to a set of energy consumers.

Each hierarchical level has a unique topology identification problem associated to it. Identification at the primary level corresponds to obtaining the set of secondary transformers

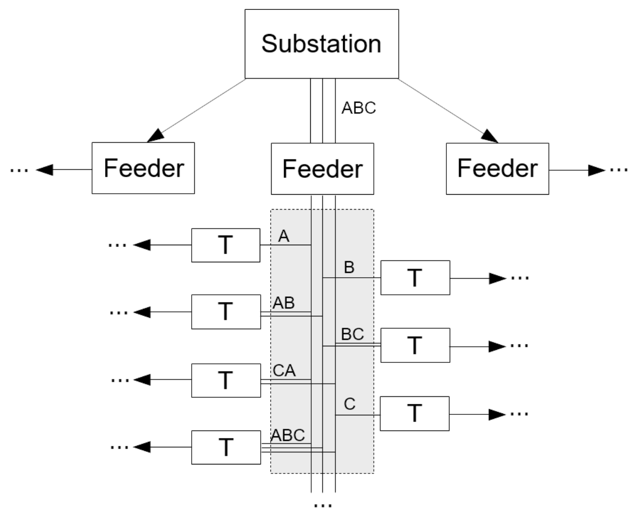


Fig. 1. Hierarchical levels of a power distribution circuit.

connected to each primary feeder, identification at the lateral level corresponds to obtaining the set of phase connections with which each secondary transformer is fed, and identification at the secondary level corresponds to obtaining the set of power consumers that each secondary transformer feeds. This paper focuses on the lateral level of topology identification: the transformer to phase connection association - also called the *Phase Identification problem*. This topology level is illustrated by the shaded region in in Figure 1.

In this paper, we will present a survey of several shallow supervised machine learning (ML) techniques applied to the Phase Identification problem. To the knowledge of the authors, such an analysis has not been performed in past literature. The authors will attempt to be as comprehensive as possible, and provide insights into why some methods work better than others.

The purpose of this paper is to act as guidance to practitioners and as a first step to researchers wishing to study the Phase Identification problem. For the practitioner, this paper will reveal which ML methods are most accurate, and include full descriptions of those methods. It will reveal what preprocessing steps are useful and/or necessary, and it will describe how to obtain an initial set of training data for high accuracy. For the researcher, it will reveal the directions of machine learning models and analysis that are most worth investigating further.

The rest of the paper is structured as follows. Section II describes and discusses past approaches to the Phase Identification problem. Section III first discusses supervised machine learning in general and then discusses, in detail, each of the machine learning methods that were tested for the Phase Identification problem. Section IV describes a method of selecting a good batch of training data to use as input for each of the algorithms. Section V discusses preprocessing transformations to the voltage data which are unique to the Phase Identification problem and describes a method for reducing the size of the dataset under consideration. Section VI presents comparisons of each technique applied to a wide range of real distribution circuits. The paper concludes with Section VII.

II. Related Work

Most phase identification research focuses on physical measurements which improve the accuracy of the periodic field testing projects themselves. Reference [1] develops a Phase Identification system based on high resolution timing measurements communicated between the base station and the feeder transformer secondaries. Reference [2] patents a method for Phase Identification through signal injection. This is the most common *field testing* method used. A signal generator is placed at the base substation and a unique signal is created for each phase. These signals are then detected by a signal discriminator at each secondary transformer. To save costs, only one signal discriminator is typically used, and it is relocated and reinstalled on a new secondary after each measurement. While fairly inexpensive, this method takes time scales on the order of several months for a whole distribution circuit. Reference [3] describes a Phase Identification technique using micro-synchrophasors. While these solutions are important, they rely on the field tests which are currently the standard operation of utility companies. As we've mentioned, these field tests are lengthy and expensive.

Some research has been done in a data driven approach to phase identification. Reference [4] predicts phase connectivity by comparing the results of a load flow analysis to measured substation voltages. This method achieved only 50% accuracy, but was improved to 78.5% by including nodal measurements of the network. This improvement, however, is highly sensitive to the locations of these added nodal measurements. Reference [5] identifies phase connectivity by finding a partition of power consumers such that the total power consumed on each phase matches the power fed by each line of the substation. There are several problems with this method. First, if there is any missing consumption data, then there will be no partition which satisfies the matching requirement. Secondly, even if a partition is found, it may not be unique - several incorrect configurations can satisfy the matching requirement. Finally, it assumes that each customer is connected to a single phase and thus does not perform well on circuits containing line-to-line connections or three phase connections. Reference [6] attempts to identify phase by computing the correlations between the voltages at the customer and lateral levels of the network. Good results were obtained, but obtaining voltage measurements at the lateral level requires extra, costly infrastructure. In reference [7] voltage magnitude data is compared to the base substation rather than the voltage at the laterals. A linear regression model is assumed to represent the measured voltage levels as a linear function of substation power on an assumed phase, substation voltage on that phase, and the power consumed by the customer. The fit with the highest R^2 value is taken. The accuracy of this method was uncertain due to model uncertainties and it is not fit for substations in delta connection or for customers that have line-to-line connections. A final track of Phase Identification research is in unsupervised learning [8] [9]. In unsupervised learning, no measurements are taken until potentially after the training phase of a given method. These methods have the potential to be much faster than supervised methods, but the use of such techniques for Phase Identification is still in its infancy.

III. Technical Methods

A. Supervised Machine Learning for Phase Identification

To perform supervised machine learning for the Phase Identification problem, we first obtain a voltage magnitude time series for each customer in the distribution circuit over some set period of time. Next, we select a small representative set of customers from the circuit to act as training data and obtain their phase connections through physical means; we will discuss the construction of this training data set in section IV, and phase labeling of this representative set is usually done via reference [2]. The goal of machine learning is to find a *constrained* function of the voltage time series which correctly predicts (or at least predicts as accurately as possible) the phase connections of this representative set. Finding this function is called *training*. The type of function returned, as well as its ability to generalize to the unlabeled data points, depends strongly on the machine learning technique used. Finally, the function is applied to every test data point's voltage time series to obtain all of the phase connections in the distribution circuit.

The main challenge with Phase Identification is keeping the size of the training data set small while still obtaining high accuracy on the rest of the customers. Since the representative set must be measured physically, the length of time needed to complete the Phase Identification problem for a given distribution circuit scales rapidly with its cardinality.

This paper focuses on discriminative models of machine learning [10]. In subsection III-B, we discuss discriminative methods in general and introduce the notation that will be used throughout the rest of the paper. The remaining subsections each describe a particular discriminative model of interest.

B. Notation

In a discriminative model, the training data $\mathcal{D}_l$ is used to learn a conditional probability distribution over the customer classes $p(t|\mathbf{x}, \mathcal{D}_l)$. This conditional distribution is then used to predict the labels of any point $\mathbf{x} \in \mathbb{R}^d$ through

$$y(\mathbf{x}) = \underset{c \in \mathcal{C}}{\operatorname{argmax}}\ p(t = c|\mathbf{x}, \mathcal{D}_l) \tag{1}$$

Which minimizes the expected classification error

$$\begin{aligned} \mathbb{E}[l(\mathbf{x}, t)] \quad &= \int \left(\int l(\mathbf{x}, y(\mathbf{x})) p(t|\mathbf{x}, \mathcal{D}_l) dt \right) p(\mathbf{x}) d\mathbf{x} \\ \text{where} \qquad l(\mathbf{x}, t) &= \begin{cases} 0 & y(\mathbf{x}) = t \\ 1 & y(\mathbf{x}) \neq t \end{cases} \end{aligned} \tag{2}$$

In most of the models we consider, learning occurs in the estimation of $p(t|\mathbf{x}, \mathcal{D}_l)$. Once this conditional distribution is estimated, the decision y always follows equation (1). This conditional probability is often parametric. When this is the case, we will denote the set of parameters in the model as θ. Parametric discriminative models are illustrated in Figure 2.

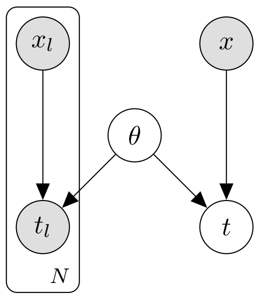


Fig. 2. Probabilistic graphical model for (parametric) discriminative methods.

In several cases, we will only be interested in a point estimate of the full conditional $p(t|\mathbf{x}, \mathcal{D}_l)$ conditioned on a particular choice of parameters $p(t|\mathbf{x}, \theta^*, \mathcal{D}_l)$ (e.g. in maximum likelihood (ML) and maximum a posteriori (MAP) methods). However, there are some techniques which use the fully marginalized conditional probability

$$p(t|\mathbf{x}, \mathcal{D}_l) = \int p(t|\mathbf{x}, \theta) p(\theta|\mathcal{D}_l) d\theta \tag{3}$$

For notational convenience, we will assume that all probability distributions are conditioned on the labeled dataset $\mathcal{D}_l$ unless stated otherwise. We will omit this explicit conditioning from now on.

We will denote the *perfect predictor function* $f^* : \mathcal{D} \to \mathcal{C}$ as the function which perfectly maps each data point (labeled and unlabeled) to its phase connectivity. We will denote the set of possible functions returned from training a model as $\mathcal{H}$, the *Hypothesis Space* of that model. We will label the set of possible hypotheses returned from an algorithm trained on a dataset $\mathcal{A} \subset \mathcal{D}$ as $\mathcal{H}_\mathcal{A}$. In general, we want $\mathcal{H}$ to contain f^*, or at least contain a very close approximation to f^*. This is true if and only if there exists a sequence of increasing training datasets $\{A_i\}_{i=1} \nearrow \mathcal{D}$, and $f^* \in \operatorname{limsup} \mathcal{H}_{A_i}$. We test this condition by training each classifier on an increasing chain of datasets and observing the accuracy on each subset. If the accuracy of the method converges to 1, then this condition is true. This last statement is sufficient but not necessary as a different chain of datasets could exist which does lead to such convergence. However, for i large enough, the differences between chains becomes negligible (as all chains must converge to $\mathcal{D}$), so if the condition is not achieved on a given chain, it is unlikely to be achieved on a different chain. On the other hand we need $\mathcal{H}_A \approx \mathcal{H}_\mathcal{D}$ for small A because we want to use as little training data as possible, and we want this to be true for *any* small A. If the sufficiency condition above is satisfied, then this means that we also want $\mathcal{H}$ to be relatively small. Finding the correct size of the hypothesis space is an implementation of the bias-variance problem [11].

C. K-Nearest Neighbors Classifier

Let K be a fixed natural number. Let $\mathbf{x}$ be a point that we wish to classify. The K-Nearest Neighbors classifier first finds the set $\mathcal{K} \subseteq \mathcal{D}_l$ containing the K points with the shortest distance to $\mathbf{x}$. This is done exhaustively via K linear searches through $\mathcal{D}_l$. Since this search must be performed for each unlabeled data point, the complexity of this algorithm is $O(K \cdot \#(\mathcal{D}_u \times \mathcal{D}_l))$ where $\mathcal{D}_u$ is the set of unlabeled data points. Approximations can be made to speed up the search [12], but this is usually unnecessary for Phase Identification because $\#\mathcal{D}_u \gg \#\mathcal{D}_l$ and the data set is not typically very large in the first place. Once $\mathcal{K}$ is found, the conditional probability of t given $\mathbf{x}$ is

$$\begin{aligned} p(t = c|\mathbf{x}) &= \frac{\mu_c}{K} \\ \text{where} \quad \mu_c &= \#\{(\mathbf{x}_l, t_l) \in \mathcal{K} \mid t_l = c\} \end{aligned} \tag{4}$$

D. Decision Trees, Random Forests, and Adaboost

A *decision tree* classifies a data point by 'asking' a series of yes or no questions about its features [13]. Each question is of the form 'Is feature m greater than value v?' The parameters m and v at each step are decided on as follows. Suppose we have already asked a series of questions and desire to ask another in a way that would help classify our dataset. We first select only training data points in which the answer to every question thus far has been 'yes'. We call this subset of training data points S. For each feature, the following score is then calculated.

$$\text{score}(m_i) = \max_v \; \{H(S) - H(S_{yes}(v)) - H(S_{no}(v))\} \tag{5}$$

where H is the sample entropy across classes, $S_{yes}(v)$ is the subset of S that is greater than v and $S_{no}(v)$ is the subset of S that is less than v. The feature m_i with the highest score is then selected (with corresponding v).

A *random forest* is nothing more than a series of decision trees. Each tree is built in the same way as above, except the feature is selected randomly instead of according to its score. The chosen value of v is still the argmax of the optimization problem in (5). Classification in a random forest is done by having each tree in the forest vote on a given data point's class.

A different set of voting trees can be formed by boosting [14]. The most popular version of boosting is called *Adaboost*. When building a new tree in Adaboost, we keep track of all training data points that the previous trees classified incorrectly and 'focus' on these points by giving them higher weights in equation (5).

The weights of each training data point are initially equal to $\frac{1}{\#\mathcal{D}_l}$. Every time a tree is built, the misclassified data points are collected. Call these points X_{miss}. The data weights of these misclassified data points are then updated by

$$w_x \mapsto w_x e^{\alpha}, \; (\forall x \in X_{miss}) \tag{6}$$

where

$$\alpha = log \; \frac{1 - \epsilon}{\epsilon} \tag{7}$$

$$\epsilon = \frac{\sum_{X_{miss}} w}{\sum_X w} \tag{8}$$

After this update, all weights are renormalized such that they sum to 1. In the final classification stage, each tree votes with weight given by that tree's corresponding α.

E. Softmax / Perceptron Classifier

In Softmax classification, the conditional distribution of class given a data vector and the parameters θ is modeled parametrically by

$$\begin{gathered} p(t = c|\mathbf{x}, \theta) = \frac{e^{\mathbf{w}_c^T \tilde{\mathbf{x}}}}{\sum\limits_{c' \in \mathcal{C}} e^{\mathbf{w}_{c'}^T \tilde{\mathbf{x}}}} \\ \text{where } \tilde{\mathbf{x}} = \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix}^T \end{gathered} \tag{9}$$

The parameters of the model are $\theta = \{\mathbf{w}_c\}_{c \in \mathcal{C}}$. These are $d + 1$ dimensional vectors. The first element of each vector represents a bias term which influences the inner product $\mathbf{w}_c^T \tilde{\mathbf{x}}$ independently from $\mathbf{x}$.

A point estimate of θ can be found through maximum likelihood. The likelihood function is

$$\begin{gathered} p(\mathcal{D}_l|\theta) = \prod_{l \in \mathcal{L}} \prod_{c \in \mathcal{C}} p_c(\mathbf{x}_l, \theta)^{t_{lc}} \\ \text{where } t_{lc} := \begin{cases} 1 & c = t_l \\ 0 & c \neq t_l \end{cases} \end{gathered} \tag{10}$$

Where we have denoted $p(t = c|\mathbf{x}, \theta)$ as $p_c(\mathbf{x}, \theta)$ for notational simplicity.

Taking the logarithm of this likelihood function and maximizing with respect to θ yields the following optimization problem

$$\underset{\theta}{\text{minimize}} \; J(\theta) = -\sum_{l \in \mathcal{L}} \sum_{c \in \mathcal{C}} t_{lc} log \; p_c(\mathbf{x}_l, \theta) \tag{11}$$

in which the objective function can be seen as a minimization of the cross entropy between $p(t_l|x_l, \theta)$ and an observed one-hot distribution t_{lc}. The optimization can be solved through gradient descent or Newton's method where the gradient and the block components of the Hessian are given by

$$\begin{aligned} \nabla_{\mathbf{w}_c} J(\theta) &= \sum_{l \in \mathcal{L}} \left(p_c(\mathbf{x}_l, \theta) - t_{lc}\right) \tilde{\mathbf{x}}_l \\ \nabla_{\mathbf{w}_{c_2}} \nabla_{\mathbf{w}_{c_1}} J(\theta) &= \sum_{l \in \mathcal{L}} p_{c_1}(\mathbf{x}_l, \theta)(\delta_{c_1, c_2} - p_{c_2}(\mathbf{x}_l, \theta)) \tilde{\mathbf{x}}_l \tilde{\mathbf{x}}_l^T \end{aligned} \tag{12}$$

F. Shallow Feedforward Neural Networks

We consider a fully connected neural network with a single hidden layer. To obtain the output class probabilities under this model, An input point $\mathbf{x}$ is first fed through an affine transformation $W\tilde{\mathbf{x}}$. The result is then passed through an element-wise nonlinear activation function σ, and that output is fed into a Softmax classifier. In symbolic notation,

$$\begin{gathered} \mathbf{z}(\mathbf{x}) = \sigma(W\tilde{\mathbf{x}}) \\ p(y = c|\mathbf{x}, \theta) = \frac{e^{\mathbf{w}_c^T \tilde{\mathbf{z}}}}{\sum\limits_{c' \in \mathcal{C}} e^{\mathbf{w}_{c'}^T \tilde{\mathbf{z}}}} \end{gathered} \tag{13}$$

The parameters of this network are the matrix W, and the vectors $\mathbf{w}_c$ $(\forall c \in \mathcal{C})$. The activation function σ is considered a hyperparameter and is not trained directly. Most recent results in Neural Networks suggest the use of the rectified linear unit as activation function [15] or one of its variants [16] [17].

Denoting the input to the nonlinearity as $\mathbf{a} = W\tilde{\mathbf{x}}$, the i^{th} component of the rectified linear unit output is given by

$$\sigma(\mathbf{a})_i = \begin{cases} 0 & a_i < 0 \\ a_i & a_i \geq 0 \end{cases} \tag{14}$$

The parameters of a neural network are estimated via maximum likelihood (11) but with the additional matrix parameter W. The optimization is estimated via gradient descent, where the gradients are calculated through a technique called backpropagation [18].

G. Bayesian Neural Networks and MC Dropout

A Bayesian Neural network [19] has the same architecture as a standard neural net, but we consider the entire posterior distribution over the parameters instead of the mode of that posterior.

$$p(t|\mathbf{x}, \mathcal{D}_l) = \int p(t|\mathbf{x}, \theta)p(\theta|\mathcal{D}_l)d\theta \tag{15}$$

Thus, if several configurations of weights predict the training data with high accuracy, then all of those configurations are taken into account when predictions are made on the remaining data. Using the full posterior may increase accuracies when low amounts of training data are used.

The trouble with this method is that the posterior $p(\theta|\mathcal{D}_l)$ is intractable for general neural networks. Thus we must use some sort of approximation scheme to implement this technique. One popular method, called *variational inference* [10], adheres to the following logic.

Consider any probability distribution over θ, $q(\theta)$. Regardless of what q is, the log posterior distribution can be decomposed as.

$$log\, p(\theta|\mathcal{D}_l) = \int q(\theta) log\, p(\mathcal{D}_l, \theta)d\theta + \mathcal{H}(q) + KL(q||p) \tag{16}$$

where $\mathcal{H}(q)$ is the entropy of the distribution q and $KL(q||p)$ is the *Kullback-Keibler Divergence* between $q(\theta)$ and $p(\theta|\mathcal{D}_l)$. Since the KL term is always positive, the sum of the first two terms represents a lower bound on the posterior. We will denote this term, called the Evidence Lower Bound (ELBO) [20], as $\mathcal{L}(q)$. To perform variational inference, we first make some assumptions about q (chosen carefully to make q tractable), and then maximize $\mathcal{L}(q)$ subject to these assumptions. The optimization occurs with respect to q or the parameters of q if we assume that q is parametric.

This maximization has two interpretations. The first is made by noting that the sum $\mathcal{L}(q) + KL(q||p)$ is constant with respect to q by equation (16). Thus maximizing the ELBO is equivalent to minimizing the KL divergence between the assumed distribution q and the true posterior distribution (which is an intractable optimization problem due to the presence of the posterior in the objective function). The second interpretation is found by looking at the terms in the ELBO. Optimizing the first term corresponds to putting high weight on parameters that explain the training data, and the second term limits this behavior by forcing q to have high entropy.

When the variational distribution is parametric, i.e. $q(\theta) = q(\theta|\lambda)$, optimization of the lower bound can be performed stochastically via black-box variational inference [21]. The gradient of the ELBO is given by

$$\nabla_q \mathcal{L}(q) = \mathbb{E}_q[(\nabla_q log\ q(\theta|\lambda))\left\{log\ p(\mathcal{D}_l, \theta) - log\ q(\theta|\lambda)\right\}] \tag{17}$$

In particular, if $q(\theta|\lambda)$ is a member of the exponential family, then every term on the inside of the expectation is tractable and can be sampled for a Monte Carlo estimate of this gradient.

A second way to obtain a full posterior is called *MC Dropout*. MC dropout still uses variational inference, but does not perform optimization through equation (17). This method is motivated by reference [22], which shows that a forward pass through a neural network containing *dropout* layers (a layer which randomly sets hidden units to zero) is equivalent to a sample of a Gaussian process approximated through a variational distribution q_{gp}. That is, if $\mathbf{p}_{dr}(\mathbf{x})$ is the result of a forward pass trough a neural network with dropout layers, then

$$p(t|\mathbf{x}, \theta) \approx \mathbf{p}_{dr}(\mathbf{x}), \theta \sim q_{gp}(\theta) \tag{18}$$

Then the full posterior can be Monte-Carlo estimated as

$$p(t|\mathbf{x}) \approx \frac{1}{S}\sum_{s=1}^{S} \mathbf{p}_{dr}(\mathbf{x}) \tag{19}$$

Thus an approximate full posterior can be obtained by performing S forward passes through a dropout network and averaging the results.

IV. Training Data Selection

Before applying any of these techniques, we need to select a set of customers S to measure as training data. Since these measurements are labor intensive, we wish for this set to be small but still satisfy $f_S \approx f^*$. We suggest a greedy submodular selection approach as described in reference [23].

First, we construct a similarity matrix A from the dataset. There are many ways to do this; the most common of which are listed below [24].

- Radial Basis Function Kernel.

$$A_{ij} = e^{\frac{-\lambda\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2}} \tag{20}$$

- Symmetrized k-Nearest-Neighbors Graph.

$$\tilde{A}_{ij} = \begin{cases} 1 & \mathbf{x}_i \in Neigh_k(\mathbf{x}_j) \\ 0 & else \end{cases}$$

$$A = \frac{1}{2}(\tilde{A} + \tilde{A}^T) \tag{21}$$

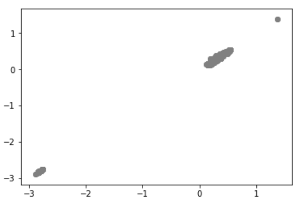
- Cosine Kernel.

$$A_{ij} = \frac{\mathbf{x}_i^T\mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|} \tag{22}$$

With the similarity matrix chosen, we use the *facility location* function $r : 2^{\mathcal{D}} \rightarrow \mathbb{R}$ to score subsets $S \subseteq \mathcal{D}$ of training data.

$$r(S) = \sum_{x_i \in \mathcal{D}} \max_{j \in S} A_{ij} \tag{23}$$

Intuitively, r is large when every data point in $\mathcal{D}$ is well represented by a training data point is S. furthermore, r has the following three properties.


(a) Untransformed

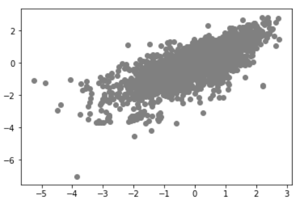

(b) Transformed

Fig. 3. Projection of data points for circuit III before and after the $120V$ stability transformation (25).

- r is submodular. If $S \subseteq T \subseteq \mathcal{D}$ and $\{\mathbf{x}\} \in \mathcal{D} - S - T$, then $r(S \cup \{\mathbf{x}\}) - r(S) \geq r(T \cup \{\mathbf{x}\}) - r(T)$.
- r is nondecreasing in $\#S$.
- $r(\emptyset) = 0$.

Such submodular functions have been studied extensively in terms of approximate optimization. In particular, it can be shown that greedy optimization of r achieves the following bound [25].

$$r(S_{\text{greedy}}) \geq (1 - \frac{1}{e}) \underset{S \in 2^{\mathcal{D}}}{max} \ r(S) \ s.t. \ \#S = K \tag{24}$$

Thus, a good training data set can be found through a greedy search over unlabeled data points.

V. Preprocessing

A. Voltage Transform

Voltage magnitude data comes in two discrete chunks; one with values near $120V$ and the other with values near $240V$. This is due to center tapped transformers on the customer side of a secondary transformer which are capable of providing either value. Most data points will have values near $240V$; only a small subset will have values near $120V$. These $120V$ points have an outlier-like effect on most supervised machine learning methods; that is, they cause significant instability in the training phases. Unfortunately, we cannot remove them as they consist of a sizable portion of customers in the distribution circuit.

We can, however, take care of these instability problems by removing the projection of each data point along the direction of the d-dimensional vector $\mathbf{1} = \begin{bmatrix}1 & 1 & ... & 1\end{bmatrix}^T$. That is, each data point is transformed through

$$\mathbf{x} \mapsto \mathbf{x} - \frac{\mathbf{1}\mathbf{1}^T\mathbf{x}}{d} \tag{25}$$

The usefulness of this transformation is illustrated in Figure 3, which shows that a dataset mapped through this transformation is much more well conditioned than its untransformed counterpart.

B. Feature Reduction

It is often the case in machine learning that removing redundant features in a dataset can lead to increased performance. In the case of Phase Identification, this means that we should remove a set of hours from each customer's voltage time series in which no novel information is contained.

For this task, we consider the dual dataset $\mathcal{D}^*$. This dataset is constructed as follows. Let X be the design matrix of the original dataset $\mathcal{D}$. That is, X is the matrix formed by stacking the row vectors $\mathbf{x}^T \ \forall(\mathbf{x}, y) \in \mathcal{D}$. Then $\mathcal{D}^* = \{m \mid m \text{ is a column of } X\}$. Each point in $\mathcal{D}^*$ corresponds to one timestamp of observation.

Next, we create a similarity matrix over $\mathcal{D}^*$ with the cosine kernel (22) and again perform greedy optimization of the facility location function (23) to obtain an approximate optimal subset of $S^* \subseteq \mathcal{D}^*$ such that every feature in the original dataset is well represented in the subset. From these selected points, we create the matrix $\tilde{X}$ whose columns are the vectors in S^*. Finally, we take the reduced feature dataset $\tilde{\mathcal{D}}$ obtained from the rows of $\tilde{X}$.

VI. Results

A. Description of Data Sets

This analysis will be performed over 7 circuits of varying complexity from Southern California Edison, Pacific Gas and Electric Company, and FortisBC. The details of these circuits are contained in the Table I. Each circuit contains 31 days of voltage magnitude data, sampled hourly for a feature vector of dimension 744.

TABLE I
Distribution Circuits Characteristics

Name	$N_{consumers}$	Phase Connections	Degree of Balance
I	1892	All	Low
II	3166	All	Low
III	4629	All	High
IV	3638	All	High
V	1310	line-line	Low
VI	358	line-neutral	Low
VII	1773	line-neutral	Low

Empirically, circuits with more potential phase connections (e.g. A, B, C, AB, BC, CA vs. just A, B and C) typically have lower Phase Identification accuracy. This is firstly due to the fact that the difficulty of a classification task is related to the number of classes, but also due to the fact that there are nontrivial dependencies between some of these classes; for example, transformers of the AB class take current from the A line and send it back along the B line, which complicates the dynamics of transformers attached to just A or B. Balanced circuits also have lower Phase Identification accuracy than unbalanced ones, but the effect is less significant. The more phase connections available and the more balanced the circuit is, the more 'difficult' that circuit is to identify.

In phase identification, we want to use as little training data as possible, so creating a validation set is impractical. On the other hand, we typically want to perform phase identification on several circuits in a given region - all of which have similar sizes and dynamics. Thus, hyperparameters that work well for one circuit in this region are likely to work for the rest. Thus, for phase identification, we replace the notion of a validation set with that of a validation circuit; i.e. we take the labels of one entire circuit and train our hyperparameters by testing (full) accuracies on that circuit. In this paper, all hyperparameters were tuned on the circuit III with a random

5% training sample. The same set of hyperparameters were used for every circuit. Thus results relating to circuit III should be considered as validation results (i.e. they carry less significance), while results relating to the rest of the circuits should be considered as out of sample results.

500 hidden units were used for the hidden layers of all neural networks and a dropout rate of 0.7 was used for MC Dropout. 100 samples were used for monte-carlo estimation of the posterior distribution. Euclidean distance was used for nearest neighbors. The random forests contained 25 trees each and 25 trees were used for adaboost as well.

Every test (except those using strategically selected training data) uses 10 random samples of the specified training portion, and the accuracy results are averaged across each trial.

B. Hyperparameters

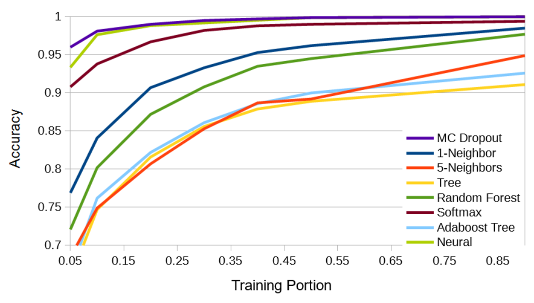
In the tests conducted in this paper, Nearest Neighbors was tested with 1 and 5 neighbors. Our binary decision trees used a maximum depth of 100, and our random forests used 10 trees with the same depth limit. We found that increasing the number of trees held the accuracy relatively constant. We tested several values of number of hidden units for and number of layers for the shallow neural network and found 500 hidden units with two layers to be sufficient; these obtained the condition for $f^* \in \limsup \mathcal{H}_{A_i}$, so more complicated models would just add variance to a problem with little bias. We found that standard L_1 and L_2 regularization did not increase accuracies on either the linear or the neural network models. However, MC dropout can be viewed as a form of regularization, so in this sense some regularization is reported. Several dropout rates were tested for MC dropout and the best performing rate of 0.1 was reported.

C. Baseline Accuracies

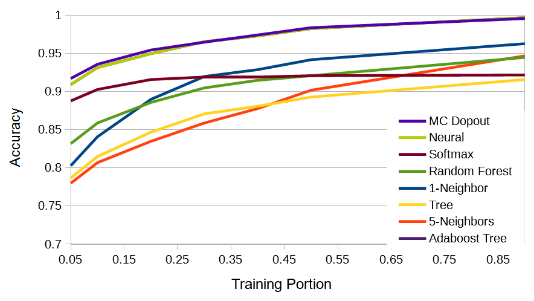
Figure 4 shows the accuracy of each method vs. the portion of (randomly) selected training data for the four circuits with all seven types of phase connections available. Figure 5 shows the same for circuits in which the number of possible phase connections is limited. We see that, as expected, the circuits with more phase connections typically achieve lower accuracy than those with limited phase connections. One interesting exception to this rule is the circuit VII, which performs fairly poorly compared to the other circuits. We suspect this accuracy drop to be due to the low amount of customers in that circuit; the low number of total customers leads to a very low total number of training data points.

In general, softmax regression and shallow neural networks have far greater accuracy than other methods, with shallow neural networks winning by a small margin in most cases. MC Dropout provides a sizable advantage over the standard neural network at low training data, and this advantage diminishes as the amount of training data increases.

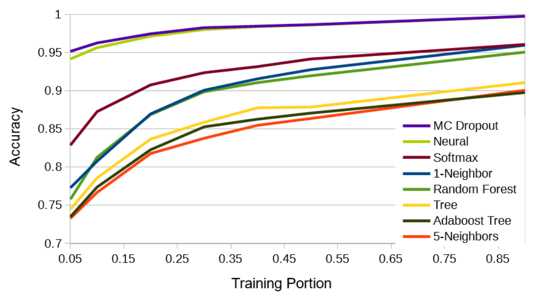
Importantly, in every circuit except I, only the shallow neural network approaches 100% accuracy as the training portion approaches 1.0. Thus, for all of these networks, we can only guarantee that f^* lives in the hypothesis space defined by the two layer neural network. Since the softmax classifier



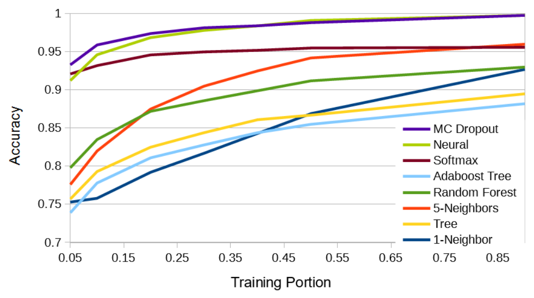
(a) I



(b) II



(c) III



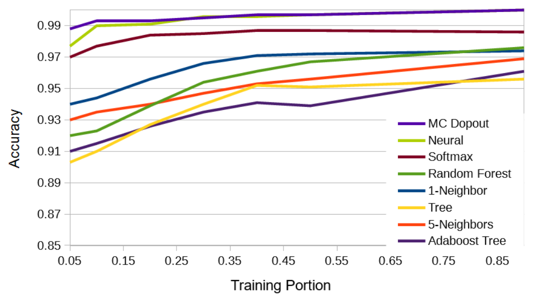
(d) IV

Fig. 4. Accuracy vs. selected training portion for circuits with all possible connections.

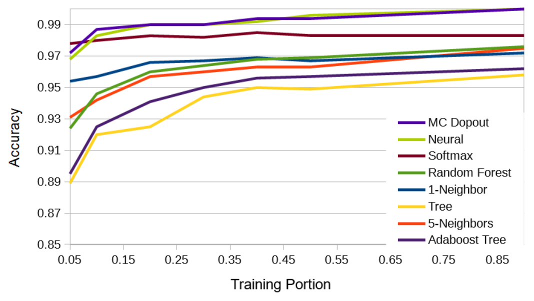
is equivalent to a single layer neural network, the increase in complexity from the softmax classifier to the two layer neural network is small, and so the latter's hypothesis space has the desired properties detailed in section III-B.

D. Training Data and Feature Selection Accuracies

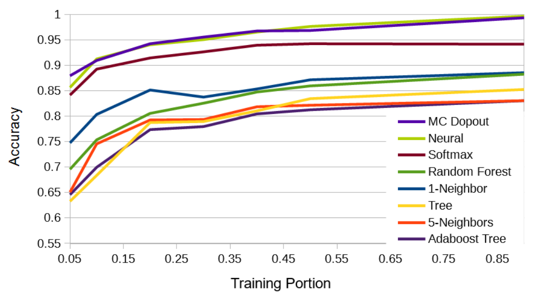
Table II shows the accuracies of the shallow neural network with MC dropout for each circuit for each of the 3 following cases: 5% *random*, 5% *selected*, and 1% *random*. The 5%



(a) V



(b) VI



(c) VII

Fig. 5. Accuracy vs. selected training portion for circuits with only phase to neutral or only phase to phase connections.

random condition is identical to the results of the previous subsection at 5% training data. Condition 5% *selected* uses the methodology of section IV to obtain a 5% training set of selected data. Condition 1% *random* is the average accuracy over ten trials of MC dropout when the training data portion is reduced to a stratified 1% sample.

For training data selection, we found that all of the listed similarity matrices worked well for circuit III, but the RBF kernel outperformed the others slightly. Thus the reported accuracies use the RBF kernel for training data selection with $\lambda = \frac{1}{\#\mathcal{D}}$.

Figure 6 shows the effect of feature reduction on the three largest circuits. The dotted lines labeled *threshold* are each 0.005 below each curves respective maximum. Interestingly, the effect of feature selection appears small. While feature selection does not improve Phase Identification accuracy, it also does not reduce it significantly until less than about 30% of the features are used. Thus the feature set used in this analysis is highly redundant, but removing this redundancy does not yield higher accuracy.

TABLE II
ACCURACY RESULTS FOR GIVEN TRAINING CONDITIONS.

circuit	5% random	5% selected	1% random
I	.934	.979	.733
II	0.910	.953	.858
III	0.952	.967	.876
IV	0.933	.972	.828
V	.988	.996	.935
VI	.880	.919	.641
VII	.972	.986	.903

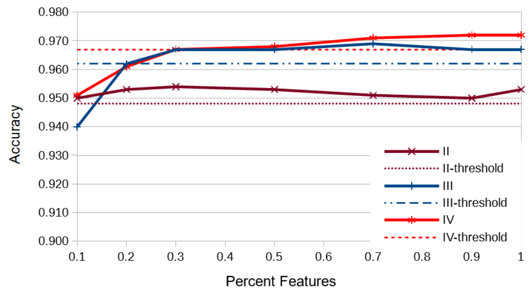


Fig. 6. Accuracy vs. selected feature portion for the three largest circuits. Dashed lines correspond to constant accuracies at each curves respective maximum minus 0.005.

Table II shows the results of training data selection. In all such cases, every class was represented before the 5% cutoff. Strategically selecting training data according to section IV yields an increase in Phase Identification accuracy for every circuit; the average increase being 3% (a 48.75% average reduction is classification errors).

Using a 1% training data sample yields an expected drop in identification accuracy. Accuracies this low are probably not useful to a utility company wishing to use machine learning for Phase Identification. Furthermore, the training data selection process is unlikely to find a representative data point of each phase connection type at such a low subset of training data. Effectively reducing the necessary training data percent to such a low value is the primary goal of future research.

VII. CONCLUSION

We have provided an analysis of shallow learning for the Phase Identification problem over a diverse set of distribution circuits. For all circuits, a 2 layer neural network with 500 hidden units and relu activation out-performs all other methods, and MC dropout provides a slight boost to accuracy over this method at low training data portions.

When the number of customers in a circuit is low (e.g. on the order of 100 instead of 1000), machine learning techniques may see lower accuracy than in circuits with more customers.

Feature selection is found to have little effect on Phase Identification accuracy. However, selection of training data according to a greedily optimized facility location function yields significantly improved and consistent results.

REFERENCES

[1] C.-S. Chen, T.-T. Ku, and C.-H. Lin, "Design of phase identification system to support three-phase loading balance of distribution feeders," *IEEE Transactions on Industry Applications*, vol. 48, no. 1, pp. 191–198, 2012.

[2] K. J. Caird, “Meter phase identification,” Mar. 27 2012, uS Patent 8,143,879.

[3] M. H. Wen, R. Arghandeh, A. von Meier, K. Poolla, and V. O. Li, “Phase identification in distribution networks with micro-synchrophasors,” in *2015 IEEE Power & Energy Society General Meeting*. IEEE, 2015, pp. 1–5.

[4] M. Dilek, R. P. Broadwater, and R. Sequin, “Phase prediction in distribution systems,” in *Power Engineering Society Winter Meeting, 2002. IEEE*, vol. 2, 2002, pp. 985–990.

[5] V. Arya, D. Seetharam, S. Kalyanaraman, K. Dontas, C. Pavlovski, S. Hoy, and J. R. Kalagnanam, “Phase identification in smart grids,” in *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, Oct 2011, pp. 25–30.

[6] H. Pezeshki and P. J. Wolfs, “Consumer phase identification in a three phase unbalanced LV distribution network,” in *2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, Oct 2012, pp. 1–7.

[7] T. A. Short, “Advanced metering for phase identification, transformer identification, and secondary modeling,” *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 651–658, June 2013.

[8] W. Wang, N. Yu, B. Foggo, J. Davis, and J. Li, “Phase identification in electric power distribution systems by clustering of smart meter data,” in *Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on*. IEEE, 2016, pp. 259–265.

[9] W. Wang, N. Yu, and Z. Lu, “Advanced metering infrastructure data driven phase identification in smart grid,” *GREEN 2017 Forward*, pp. 16–23, 2017.

[10] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[11] Y. Le Borgne, “Bias-variance trade-off characterization in a classification problem: What differences with regression,” *Machine Learning Group, Univ. Libre de Bruxelles, Belgium*, 2005.

[12] K. Hajebi, Y. Abbasi-Yadkori, H. Shahbazi, and H. Zhang, “Fast approximate nearest-neighbor search with k-nearest neighbor graph,” in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 1, 2011, p. 1312.

[13] W.-Y. Loh, “Classification and regression tree methods,” *Encyclopedia of statistics in quality and reliability*, 2008.

[14] B. P. Roe, H.-J. Yang, J. Zhu, Y. Liu, I. Stancu, and G. McGregor, “Boosted decision trees as an alternative to artificial neural networks for particle identification,” *Nuclear Instruments and Methods in Physics Research A*, vol. 543, pp. 577–584, May 2005.

[15] S. Sonoda and N. Murata, “Neural network with unbounded activation functions is universal approximator,” *ArXiv e-prints*, May 2015.

[16] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (ELUs),” *ArXiv e-prints*, Nov. 2015.

[17] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, “Self-normalizing neural networks,” *ArXiv e-prints*, Jun. 2017.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[19] J. Lampinen and A. Vehtari, “Bayesian approach for neural networksreview and case studies,” *Neural networks*, vol. 14, no. 3, pp. 257–274, 2001.

[20] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, “Variational inference: A review for statisticians,” *ArXiv e-prints*, Jan. 2016.

[21] R. Ranganath, S. Gerrish, and D. Blei, “Black box variational inference,” in *Artificial Intelligence and Statistics*, 2014, pp. 814–822.

[22] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in *International Conference on Machine Learning*, 2016, pp. 1050–1059.

[23] H. Lin and J. Bilmes, “How to select a good training-data subset for transcription: Submodular active selection for sequences,” Washington University Seattle Dept. of Electrical Engineering, Tech. Rep., 2009.

[24] U. Von Luxburg, “A tutorial on spectral clustering,” *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[25] A. Krause and D. Golovin, “Submodular function maximization.” 2014.