Machine Learning Engineer Nanodegree

Capstone Proposal

Yves Fauser July 3rd, 2019

Proposal

Domain Background

Overhead power lines are susceptible to various outside influences like tree branches falling onto the line, mechanical damage through material fatigue, etc.

Failing power lines can lead to power outages, cutting of whole rural villages and cities from electricity supply, or even worse, line faults can cause fires through heat caused by power resistance and discharges.

The problem, and how it can be solved with the help of machine learning, is detailed in an academic paper written by the technical University of Ostrava (Czech Republic) [2].

Problem Statement

Partial damage of overhead power lines by external factors is hard to diagnose. Even with large scale regular inspection of power lines e.g. through manual visual inspection, or fly overs with drones, a lot of material fatigue and partial damage problems can stay undiscovered. Therefore a solution is needed that relies solely on analysis of sensor data, discovering noise patterns of signals measured at the overhead power line. A key early indicator of a condition of the overhead power line that needs attention is partial discharge (PD). PD is generating specific patterns that can be seen in the sensor data, and gives a good indication e.g. of tree branch contact.

[1] `Whenever partial discharge is initiated, high frequency transient current pulses will appear and persist for nanoseconds to a microsecond, then disappear and reappear repeatedly as the voltage sinewave goes through the zero crossing`

The problem of discovering damages on overhead power lines can therefore be solved by analyzing the sensor data, finding PD patterns and distinguish them from other patterns e.g. caused by corona discharge and other commonly found external influences to the power line.

Datasets and Inputs

As part of a Kaggle competition, the Technical University of Ostrava (Czech Republic) provided sensory data of medium voltage power lines [3]. The data was obtained from a low cost sensor (metering) device developed at the University and mounted at 20 different location within a region in the Czech Republic. The data was gathered from a real life power distribution network, and is therefore noisy because of real world disturbances caused by

the power lines acting as antennas for electromagnetic signals generated in proximity of the power lines like radio stations, nearby lightning strikes and corona discharges.

The dataset is divided into one training and one test dataset available in the Apache Parquet format. Other than is most datasets each column instead of each row is containing one signal. Each signal (each column) has 800,000 measurements taken during a 20 millisecond time windows which equates to one complete grid cycle of the 3 phases measured.

The targets where manually classified by a domain expert that looked at noise patterns present in the data. If the signal is showing a PD pattern, the target is set to 1, if any other or no disturbance pattern is preset the target is set to 0. Each of the 3 phases is treated independently, so a PD pattern might be present on one of the phases, but might not be present in another. Only the training dataset is labeled with targets. The test dataset is part of the Kaggle competition and the targets are therefore hidden from public.

The data contains:

- **id_measurement:** the ID code for a trio (3-phases) of signals recorded at the same time.
- **signal_id:** A foreign key for the signal data. Each signal ID is unique across both train and test, so the first ID in train is '0' but the first ID in test is '8712'.
- **phase:** the phase ID code within the signal trio. The phases may or may not all be impacted by a fault on the line.
- **target:** 0 if the power line is undamaged, 1 if there is a fault.

The data is highly imbalanced, as it contains only 96 true positives in more than 10,000 raw signals.

Solution Statement

The PD patterns found in the sensor data gives an indication of the target fault condition (tree branch fallen onto the line, material fault, ground contact). AI models can be used to distinguish the patterns found in the data like radio transmission interference, random impulses, etc. from the PD patterns.

Benchmark Model

The data is part of a past Kaggle competition that attracted 1,451 teams to date. The leader-board contains the scores of those teams with a high range of scores starting from a low negative number up to the top score of 0.71899 [5]. In addition to the Kaggle leader-board, the Technical University of Ostrava also published results in a academic paper that is based on the work they have done using a Random Forest algorithm as the classifier [2].

The Kaggle competition are evaluated using the Matthews correlation coefficient. The results discussed in the academic paper are detailing the achieved Accuracy, Precision, Recall and F1-Score.

I will use the results detailed in the academic paper in table 4, column 'SOMA / RA,SP,HA' as a benchmark for my model. Those are:

- **accuracy**: 93.2
- **precision**: 95.2
- **recall**: 68.4
- **F1-score**: 79.1

My goal is to achieve similar or better results using a neural net than the results achieved using the Random Forest approach.

As a secondary goal, I will compare my results to the results achieved by the teams in the Kaggle competition. I'd like to be in the mid-range of the leader-board.

Evaluation Metrics

I will evaluate my work with Accuracy, Precision, Recall and F1-Score to generate results that can be compared with the results achieved by the random forest approach details in the academic paper [2].

In addition, I will evaluate my work with the Matthews correlation coefficient (MCC) between the predicted and the observed response. The MCC is given by:

$$MCC = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

where TP is the number of true positives, TN the number of true negatives, FP the number of false positives, and FN the number of false negatives.

I will use my MCC results to compare my work with the leader-board of the Kaggle competition.

Project Design

As a first step, I will have a close look at the data and build a couple of visualizations. I need to first understand what makes up a clear PD pattern in the data, and how a PD patterns differs from other measured patterns.

One possible next step is to work on filtering methods to remove non-PD patterns that are easy and safely removable from the data. However, this might be rather counter productive if it also removes PD patterns by accident.

Then I will work on extracting and/or calculating the most relevant features from each signal. Each signal has 800,000 data-points and therefore can't simply be used as the input to a Neural Net. Also, I will have to look at normalization to make sure there's not too much variance in the data.

Another detail I will have to look at is the fact that the dataset is highly imbalanced. Besides the use of the already appropriate MCC evaluation, I will potentially also need to look at under- or over-sampling of the data to fight the imbalanced nature of the dataset.

I will then divide the resulting training dataset with feature reduction into a train, validate and test dataset.

My plan and intuition is that a neural network will most likely lead to the best results, therefore I will design a neural network with the appropriate layers. I will experiment with fully connected and CNN based neural networks and use grid search to find the optimal set of hyper-parameters.

For both the fully connected neural networks and CNN I will use Normalization Layers, Dropout and Relu activation.

As for the CNN based model I will experiment with using two 1D convolutional layers, one convolutional layer in the time domain and another in the frequency domain. Those convolutional layers will then be connected to two max pooling layers whose outputs are then fed into a single final fully connected layer. This network setup has shown some good results when applied to noisy audio samples in speech recognition tasks [7].

I will also spend some time searching for pre-trained models that I could use (transfer learning). There should be pre-trained models available that should perform well in the frequency analysis domain (radio theory, etc.), e.g. there is one example on mathworks.com talking about the use of AlexNet and GoogLeNet to analyze electrocardiogram (ECG) signals which might be a good choice [6].

Finally I will run the Kaggle test dataset through my trained model and submit the results to Kaggle to get my score.

Links to resources and papers

[1] Wikipedia entry for Partial Discharge (PD):
https://en.wikipedia.org/wiki/Partial_discharge

[2] Technical University of Ostrava academic paper:
https://ieeexplore.ieee.org/document/7909221

[2] Technical University of Ostrava academic paper - Dropbox Link:
https://www.dropbox.com/s/2ltuvpw1b1ms2uu/A%20Complex%20Classification%20Approach%20of%20Partial%20Discharges%20from%20Covered%20Conductors%20in%20Real%20Environment%20%28preprint%29.pdf?dl=0

[3] Kaggle vsb-power-line-fault-detection data: https://www.kaggle.com/c/vsb-power-line-fault-detection/data

[4] Kaggle vsb-power-line-fault-detection key Intro: https://www.kaggle.com/c/vsb-power-line-fault-detection/discussion/75771

[5] Kaggle vsb-power-line-fault-detection leader-board: https://www.kaggle.com/c/vsb-power-line-fault-detection/leaderboard

[6] Classify Time Series Using Wavelet Analysis and Deep Learning:
https://www.mathworks.com/help/wavelet/examples/signal-classification-with-wavelet-analysis-and-convolutional-neural-networks.html

[7] Time-Frequency convolutional networks for robust speech recognition: https://www.sri.com/sites/default/files/publications/time-frequency_convolutional_networks_for_robust_speech_recognition.pdf