VSB Power Line Fault Detection

Abhishek Ranjan, April 11, 2019

Background - Competition

- Competition Name: VSB Power Line Fault Detection: Can you detection faults in above ground electrical lines?
- Team Name: Abs_ran
- Private Leaderboard Score: 0.70232
- Private Leaderboard Place: 4
- Name: Abhishek Ranjan
- Location: Toronto, Canada
- Email: abhishekrnjn@yahoo.com

Background - Personal

I am a machine learning researcher, specializing in biometrics research. I have a Ph.D. in Computer Science. I do not have any experience in electrical engineering, but I have extensive experience in pattern recognition in biosignals.

I found the problem of power line fault detection interesting and dataset unique due to sparsity of positive samples in extremely long time series data.

I spent approximately 50 hours on research and coding, but my computers spent many more hours training and optimizing the models.

Summary of Solution

The approach has the following highlights:

- XGBoost model [2]
- Pyramidal and multiscale 1-D Local Binary Pattern features [1]
- Artificial training sample expansion using the given training samples
- Numba on a multi-core PC for fast feature extraction

Feature Engineering

Based on the description of fault signals in Tomas Vantuch's Ph.D. thesis referenced in the competition, there were two main aspects of a fault signal: (1) shape of the waveform, (2) location/frequency of the waveform in the 800K time-series. 1-D local binary pattern (LBP)

describes waveform shapes and frequency very effectively using histogram of binary representations (see [1]). When they are calculated at multiple scales, they encode many different shapes over varying time lengths. There are, however, two main limitations of classic LBP that will limit its potential application to this problem.

1. LBP disregards the magnitude of differences in samples and only encodes the sign. I addressed this limitation by calculating multiple LBPs at different difference thresholds.
2. LBP does not encode the location of patterns, but only frequency. To encode the location of the fault pattern in the feature vector, I calculated pyramidal 1-D multiscale LBP.

Addressing the two issues resulted the following LBP configuration:
2^8 histogram bins x 6 scales or radii x 3 difference thresholds x 5 pyramidal segments x 3 phases = 69120 dimension LBP feature

Training Method

The best performing model is a boosted classifier with a very large number of learners (25,000 learners). The solution uses XGBoost library for gradient boosting. LBP features were generated from each phase signal (from 800,000 samples) separately and concatenated to create a single feature vector of approx 69K size (see previous section).

Because the number of faulty signals was small and the two classes were heavily imbalanced, I developed an algorithm to randomly perturb a feature vector to generate many more feature vectors, thus significantly increasing the positive training samples and eliminating the class imbalance for training.

XGBoost with very slow learning rate and large number of learners played an important role in avoiding overfitting.

Interesting Findings

Some of the key findings of the experiment are:

1. Pyramidal LBP exhaustively cover both frequency and time domain features.
2. Training data expansion trick helped in balancing the training classes, allowing the classifier to use large number of learners without overfitting.
3. Interestingly, a vanilla adaboost classifier with hundreds of learners trained on a high dimensional LBP feature performed reasonably well. This indicates that the features, in general, had strong discriminative power.

Simple Features and Methods

- LBP features have many parameters that can be tuned and few of the parameter sets produced an accuracy level within 10% of the best accuracy
- One recommended simplified solution: using word length of 8 and at 6 different scales (radius) resulting in 4608 dimensional feature and XGB with num_boost_round = 1000 will be significantly faster to train.

Model Execution Time

Details of the best performing solution:

- Feature extraction resource usage for the entire training set (no GPU):
    - Input: raw train.parquet file
    - Output: np.ndarray of shape (8712, 23040)
    - Total time: 6.1 hours running 20 threads
    - Processor details: Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz (24 cores) with 16GB RAM
- Training resources (no GPU):
    - Input: np.ndarray of shape (8712, 23040)
    - Output: XGBoost boosted classifier
    - Total time: approximately 24 hours (running 12 threads)
    - Processor details: Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz (8 cores) with 24GB RAM
- Testing resources (for the entire test set, no GPU):
    - Total time for feature extraction:
        - Input: test.parquet file
        - Output: np.ndarray of shape (20337, 23040)
        - Time: 14.4 hours
        - Processor: Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz (24 cores) with 16GB RAM
    - Total time for classification
        - Input: np.ndarray of shape (20337, 23040) and XGBoost booster object
        - Output: submission csv file
        - Time: 336 seconds (~6 minutes)
        - Processor: Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz (24 cores) with 16GB RAM

References

1. Multiresolution Gray Scale and Rotation Invariant Texture Classification with Local Binary Patterns, http://vision.stanford.edu/teaching/cs231b_spring1415/papers/lbp.pdf
2. XGBoost, https://xgboost.readthedocs.io/en/latest/