Interpretable Dimensionally-Consistent Feature Extraction from Electrical Network Sensors

Laure Crochepierre, Lydia Boudjeloud-Assala, Vincent Barbesant

Interpretable Dimensionally-Consistent Feature Extraction from Electrical Network Sensors

Laure Crochepierre[1,2] ✉, Lydia Boudjeloud-Assala[1], and Vincent Barbesant[2]

[1] Université de Lorraine, CNRS, LORIA, F-57000 Metz, France
{laure.crochepierre, lydia.boudjeloud-assala}@univ-lorraine.fr
[2] Réseau de Transport d'Electricité (Rte) R&D, Paris, France
{laure.crochepierre, vincent.barbesant}@rte-france.com

Abstract. Electrical power networks are heavily monitored systems, requiring operators to perform intricate information synthesis before understanding the underlying network state. Our study aims at helping this synthesis step by automatically creating features from the sensor data. We propose a supervised feature extraction approach using a grammar-guided evolution, which outputs interpretable and dimensionally consistent features. Operations restrictions on dimensions are introduced in the learning process through context-free grammars. They ensure coherence with physical laws, dimensional-consistency, and also introduce technical expertise in the created features. We compare our approach to other state-of-the-art feature extraction methods on a real dataset taken from the French electrical network sensors.

Keywords: Grammar-Guided Genetic Programming (GGGP) · Supervised Learning · Feature extraction · Interpretability · Electrical Power System

1 Introduction

Electric transmission power grids are large complex systems monitored and operated in real-time, 24/7, by highly trained control room operators (also called dispatchers). Their task is mainly to ensure that the overall system, critical in modern societies, remains in a secure state at all times to conduct electricity from producers to consumers. In particular, they watch over the *electrical flow* on each line to keep it under its *thermal limit*; a physical threshold above which a short-circuit could happen, risking for the safety of property and people nearby. To accomplish this monitoring, a large number of sensors placed throughout the electrical network provide them measurements relayed by a large number of screens in the control room. From these measurements, operators continuously perform information synthesis to prepare their strategy upstream and plan preventive actions (mainly changing the network topology) to redirect the power flow before it reaches its limit. However, even if their ability to run the power system is well established, highlighted by the absolute absence of any significant blackout recently, Transmission System Operators (TSO) have noticed a steep

rise in the complexity of real-time operations [8]. This trend is mainly linked to market dynamics, increased renewable energy sources connected to the grid, and the development of electrical interconnections with other European countries. As a consequence, power lines are operated closer to their thermal limit, and dispatchers have to go through their decision-making process faster to keep time to handle more critical situations. Today, the information synthesis step is computer-assisted by some hand-crafted aggregation indicators and computationally massive simulations calculated from the network measurements. Historically created by operators with the use of their expert knowledge, these few indicators aren't exhaustive and can not confirm the safety of all situations regarding electrical flows. Also, these indicators might need to be revised more frequently than they are now, given the system's current dynamics (e.g., newly installed renewable power plants or cross-border flow thresholds adjusted by the markets). Besides, the simulations are quite long to compute and cannot cover all possible forecasts of the future. Consequently, operators still perform some parts of this information synthesis by themselves using their knowledge of the system and the outputs of the simulations to synthesize measurements, results, and information about the connection of the lines in the grid.

A recent study on the French electrical transmission power system [31] proposed an exploratory dimensionality-reduction method, to identify interactively some factors influencing atypical consumption behaviors. In this experiment, knowledge was introduced by conditioning autoencoder with input features experts thought to be causing the output behavior. As in our application data is given with different physical dimensions such as voltage and active power and needs to respect physical properties, we were interested in finding ways to inject a different kind of knowledge coming from the field of physics (Ohm's law for example). Therefore, the aim of our work is to propose an automatic feature extraction method to *explain* electrical flow with physically consistent and intelligible indicators created from sensor data. From this point on, dispatchers could directly then use these indicators as a surrogate of the status of the power network zone they look after.

In this context, we investigate how to perform *feature extraction* with expert constraints for power line flow explanation, by creating relevant and *potentially non-linear combinations* of features from the initial dataset. The proposed approach relies on Grammar-Guided Genetic Programming [34], often abbreviated as GGGP (or G3P), to extract human-readable combinations of features. Our contribution is twofold. First, we propose a feature creation method which integrates domain-knowledge from power system experts using a context-free grammar build interactively with them. More specifically, this grammar includes some physical properties of electrical systems and prevents from using worthless combination operators, which helps reduce the search space. Finally, the created features are analyzed by a human expert who provides insights on what is correct and what would be expected from operators. The second contribution presented in the experiments is the interpretability evaluation of the outputs. Following the terminology used by Doshi-Velez and Kim in [9], we performed both "Functionally-Grounded"

and "Human-Grounded" evaluations of the proposed approach by comparing it to other interpretable state-of-the-art methods and give created feature for expert analysis. In this paper, we also use a correlation-based optimization objective as a metric to evaluate individuals. We compare this metric to distance-based metric Mean Squared Error (MSE) to detail why both metrics can't be used equivalently. Throughout this article, we use the real-world dataset created from measurements of the French power grid, from January 2014 to December 2018 at 5-minute intervals.

This paper is organized as follows. Section 2 summarizes related works on the topics of feature extraction, interpretability, and Grammar-Guided Genetic Programming. Then, Section 3 details the data used to develop our method. In Section 4, we described the proposed approach. The experimental evaluations and their results are presented in Section 5, and finally, Section 6 concludes this paper and introduces some future works.

2 Related Works

2.1 Feature Extraction

Real-world applications often produce data in a very high-dimensional space [25], but they are very sparse, redundant, and their underlying structure is often representable in a much lower dimension. In this context, dimensionality reduction (DR) techniques can be of great support to visualize data or improve a classifier performance [49] and are even used for data compression [6]. Thus, DR is an important preprocessing step in many machine learning pipelines. More formally, DR can be defined as the set of techniques taking inputs X with a high number of features D, and mapping it to a reduced set of features X' with size d, such as $d << D$ while retaining as much information from the original structure as possible. Exhaustive reviews on this topic can be found in [17, 27, 47].

DR is mainly done using two types of approaches: feature selection or feature extraction (also called sometimes feature transformation, augmentation, or creation). While feature selection only selects the most informative features from the dataset, feature extraction tries to effectively *combine features* from the original dataset to produce more expressive ones. Among the feature extraction methods, another distinction can be made between linear and non-linear methods (also called manifold-learning methods). Linear methods have been used for a long time. They include methods such as Principal Component Analysis (PCA) [16] which finds axis by variance maximization, Laplacian Eigenmaps [3], Non-Negative Matrix Factorisation [24] or Locally Linear Embedding (LLE) [36]. PCA has the advantage of providing quite interpretable results using the selected principal components [48]. However, in many cases, the data structures are too complex, and linear mappings cannot retain enough information from the initial feature space. In this context, non-linear mappings are considered to represent the original data as closely as possible. Among non-linear algorithms, Isomap [46] is one of the widely used methods. Other methods include Kernel PCA [43] (non-linear extension of PCA), t-SNE [30] or UMAP [33] two dimensionality-reduction

methods for data visualization, or deep learning methods such as Variational Autoencoders [18] or Conditional Autoencoders [45].

However, these DR methods were initially presented in an unsupervised setting and did not use any supervision scheme. This element is an issue in our application, as they can't take into account the valuable knowledge of available target values. More recent works focus either on how to extend classical methods to supervised configurations (supervised-LE) [38] or on how to take advantage of some target features to structure the new feature space: for example by integrating an additional optimization objective (i.e., as second loss term in the neural network [28]) or producing multiple transformations one for each class.

2.2 Interpretability

One of the recurring concerns about DR is the lack of interpretability of the axes in the feature space. As we discussed above, linear methods are often considered as interpretable (even when they add up different dimensions), but it is not the case for non-linear ones. It has been observed that many DR methods construct the new feature space "upon arbitrary combinations of many uncorrelated physical dimensions" [15], leading to the non-usability in many industrial processes. Some promising works propose interpretable DR methods based on kernel dimensionality reduction [50, 15], which are able to project the embedding dimensions on the label-space to make interpretations.

In the supervised machine learning community, the interpretability of the results is a key challenge to improve the user *trust and acceptation* of the created model, and the proposed results. The existing interpretability methods are roughly divided into two categories: interpretable models and post-hoc interpretability. Interpretable models include Linear Regression, Decision Trees [5], Generalized Additive Models [13], or Rule Fit [12]. They produce interpretable outputs, but they are often considered as sub-optimal regarding complex classification or regression tasks. On the other side, post-hoc interpretability is often used for more complex models which produce more accurate results such as Deep Neural Networks. This problem is called the *accuracy-interpretability trade-off* [4]. With the omnipresence of deep learning models, several works focus today on the post-hoc interpretation of models using model-agnostic methods such as LIME (Local Interpretable Model agnostic Explanations) [40] or model-specific ones SHAP (SHapley Additive exPlanations) [29].

However, recent works by Laugel et al. [23] warns about the "risk of having explanations that are a result of some artifacts learned by the model instead of actual knowledge from the data". They also suggest that further research needs to be done to provide satisfying post-hoc explanations, both faithful to the predictor and to ground-truth data. Rudin et al. [41] rather suggests to start with interpretable models and only shift to black-box models if no sufficient solution has been found. They also suggest asking for *strong* explanations of the created models. Interpretability is therefore increasingly required, whether for safety, fairness, specification issues, or scientific understanding. However, there is no complete consensus for now about how to define and how to evaluate

interpretability. To answer this problem, Doshi-Velez and Kim [9] proposed a three-level evaluation: the first one is an "Application-grounded" evaluation where humans evaluate interpretability on an exact application task; the second, called "Human-grounded", uses a similarly applied evaluation but on a simpler task; finally, the "Functionaly-grounded" category uses an even simpler evaluation on a simple task and involves no human. This final category assesses, for example, multiple interpretable algorithms on the same metric to identify which one performs best. We'll detail interactivity in our experiment using this taxonomy.

2.3 Grammar-Guided Genetic Programming

Recently there has been new application perspectives for Genetic Programming (GP) regarding the increasing need for interpretable results. GP was for example used to provide interpretable policies in reinforcement learning [14], to learn manifolds [25], to create visualizations [26] or to explain complex deep learning models [10]. For dimensionality reduction tasks, GP has also been used a lot as a feature construction method [35]. It presents some advantages in comparison with other methods presented in Section 2.1. As identified in [25]:

- they try building a global learner unlike local methods such as t-SNE [30],
- they do not require a differentiable fitness function (unlike Autoencoders) and thus can be used with a great variety of objective functions,
- they intrinsically produce an interpretable mapping.

Genetic programming was initially introduced by Koza [20, 21], who identified that many problems could be reformulated as program induction. Unlike Genetic Algorithms, which evolves a population of fixed-length binary vector, GP evolves a population of programs represented as *trees*. Each tree consists of a combination of initial features using several operations taken in a list of allowed functions (e.g., $+$, $-$, $\times$, $\%$). Initial features are represented in leaves and functions in nodes.

Nowadays, there are many different variants and implementations of GP for program creation. The three major ones are the following : the first and most classical approach, tree-like GP; the second Linear GP [2] represents programs as linear sequences to perform imperative program evolution; the third is grammatical evolution [42], where the representation language uses a Backus Naur Form grammar [19] and programs-trees are derived from this grammar.

As in some cases, the search space in GP may be too large, thus preventing the algorithm from converging, some alternatives have been proposed among grammatical evolution strategies. For example, in Grammar-Guided Genetic Programming (GGGP) [22], the search space is constrained by a set of *rules* to create features. These rules are defined using grammar written in Backus-Naur Form (BNF) [19] as the one provided in Figure 1. A comprehensive review of Grammar-Guided Genetic Programming can be found in [34]. GGGP has been identified as a way to enforce expert domain knowledge into the learning procedure [39] using *ontologies*. For instance, it has been used as a way to impose constraints on the *dimensions* of variables in a classification problem [7]. This is what led us to consider this method to find explanatory variables.

3 Data Description

The electrical power network can be represented as a graph $G = (N, L)$ with a set of nodes N and lines L. Lines represent here electrical transmission power lines, and nodes are the locations where lines can be physically connected. Measurements are acquired in nodes and line extremities. In each node $n \in N$ the observed quantities are active power p_n and reactive power q_n while for a line l measures contain information about the line connection $connected_l^{or}$, $connected_l^{ex}$ and also an $neighbor\_id_l$ key corresponding to the list of neighboring lines at this timestep.

Using these measurements as inputs, simulations can be done to estimate voltage magnitude v_n and angle θ_n in nodes $n \in N$ using Netwon-Raphson based power-flow analysis [44] and eventually compute flow values at each line extremities (with origin *or* and extremity *ex*), i_l^{or} and i_l^{ex}, $\forall l \in L$. In our case, a solver computes these quantities for each timestep. In the rest of this article, flow values i_l^{or} and i_l^{ex} will constitute the output of our different methods and experiments. We can thus consider the studied system as a *closed system* without time dependency as the target features $i_l^{or,ex}$ provided by our simulator only depends on measures and expert hyperparameters setting used to calibrate the power-flow calculus.

First, we exhaustively describe the graph with variables reflecting the different electrical links ($connected_l^{or,ex}$ a boolean representing the connection of line l at its origin or its extremity, and $neighbor\_id_l$ an id used as a key to represent all lines electrically connected to line l). We move the measured variables from the nodes of the graph to each origin/end of the line connected to this node. Although this representation implies redundancy in the data, we can now reason only in terms of power lines and forget about nodes objects. From now on, we'll refer to measured and simulated variables by $X = (X_l)_{l \in L}$ and target flow variables $y = (y_l)_{l \in L}$ where :

$$\begin{aligned} \forall l \in L, \;\; & X_l = ((p, q, v, \theta)_{n\_or(l), n\_ex(l)}, connected_l^{or,ex}, neighbor\_id_l) \\ & y_l = (i_l^{or}, i_l^{ex}) \end{aligned}$$

In our case, we focus on the network operated by the French TSO called Rte (Réseau de Transport d'Electricité). The French power grid as a whole is a very complex system, with up to 6500 nodes, 12000 power lines, and many interactions both with other European networks and within it. A common approach, to control how these interactions influence studies of the grid, is to divide into sub-zones within which the elements have a high-mutual influence on each other [32]. This approach, historically done by TSOs, allows several operators to work simultaneously on separated zones of an acceptable size they can control. Thus, we focus on a specific mountainous valley where the escarpment intrinsically constrains interactions with the rest of the network. This selection restricts the study perimeter to 69 nodes and 92 lines, where 9 lines connect the zone with other parts of the network. By collecting measurements from January 2014 to December 2018, we obtained 365 165 timestep observations and target flow variables.

4 Proposed Approach

As explained in the introduction, we are interested in finding *"explanations"* about flow variable y_l, $\forall l \in L$. More formally, given a set of observed features $X \in \mathbb{R}^D$, we want to extract relevant and *potentially non-linear combinations* of these features $\forall l \in L, X_l^{prime} \in \mathbb{R}^d$ with $d << D$ and so that X_l is relevant to explain y_l. We chose to focus on Grammar-Guided Genetic Programming (GGGP) methods to propose a custom grammar for electrical data and a new correlation-based metric.

4.1 Grammar Description

Grammar construction plays a crucial role in GGGP methods as it defines the search rules in the feature space: a too-loose grammar would have a too-wide space to search, while a too-constraint grammar would be limited to sub-relevant zones. For the sake of comprehension, we provide a simplified version of our grammar in Figure 1. The complete version of the grammar uses all features described in Section 3 including topological variables, and a wider variety of functions on each dimension. The grammar is iteratively constructed with experts-in-the-loop withdrawing or adding constraints such as new variables or new operations.

Dimensions Definition Firstly, we need to define the dimensions the grammar can handle. The physical dimensions taken into account in this grammar can either be active power p (with dimension : watt W), reactive power q (volt-ampere reactive VAR), apparent power s (volt-ampere VA), voltage magnitude v (voltage V) or intensity i (ampere A). From there, we can define the square of all dimensions: $p2$, for example, is the squared value of p with dimension W^2.

Grammar Structure The first step to build a grammar is to define the output structure, here `<expr>`. The output can be one of the elements separated by the character "|". This formulation allows to enforce expert knowledge on the output dimension. In this example, `<expr>` can be of dimension q, p^2, q^2 or v^2.

The second step is to *impose* the dimensional consistency of the output variable, by defining which operations can be performed on each dimension and how to combine two dimensions. To jump from one dimension to the next, laws of physics such as the power triangle $s = \sqrt{p^2 + q^2}$ or some variation of Ohm's law $i = \frac{s}{v}$ have been expressed in the grammar. Input variables are defined here using observations `<p_var>`, `<q_var>`, `<v_var>`, and `<i_frontier_var>` ($i\_frontier$ correponding to the flow for 9 cross-border lines to model interactions with outside the zone) with corresponding dimension p, q, v, i.They can be combined to produce a new variable with either the same dimension (for example `<p> - <p>` produces a new variable with dimension p), or a different dimension (`<p>/<p>` has no dimension while `square(<p>)` has dimension p^2).

Finally, we define licit operations to perform on a single or a pair of variables, which are repeated for almost all dimensions.

```
# 1) Create unitary expressions (allowed returned dimensions)
<expr> ::= <p> | <s> | <i> | <f> * <expr> | <p>/<v>
# 2) Define legal operations on each dimension
<p>  ::= <p>-<p> | <pop>(<p>, <p>) | <sop>(<p>) | <p_var>
<q>  ::= <q>-<q> | <pop>(<q>, <q>) | <sop>(<q>) | <q_var>
<v>  ::= <v>-<v> | <pop>(<v>, <v>) | <sop>(<v>) | <v_var>
<p2> ::= <p>*<p> | square(<p>)
<q2> ::= <q>*<q> | square(<q>)
<s>  ::= sqrt(<p2> + <q2>) |<v> * <i>
<i>  ::= <s>/<v> | <pop>(<i>,<i>) | <sop>(<i>) | <i_frontier_var>
<f>  ::= <f>*<f> | <p>/<p> |  <q>/<q> | <v>/<v>
# 3) Define operations returning variable with the same dimension
<pop>::= sum | minimum | maximum # Functions with two arguments
<sop>::= abs | neg | pos # Functions with only one argument
```

Fig. 1. Grammar example. "|" represents separation between each possibility of replacement of the element located at the beginning of the line, before " ::= ". Input variables are `<p_var>` `<q_var>`, `<v_var>` and `<i_frontier_var>`.

4.2 Methodology Description

We base our method on GGGP algorithms and introduce human-experts in the grammar construction process. First, as in GGGP methods, we use a population-based search on the space by performing multiple times operations (selection, crossover, mutation) on a group of individuals, as described in Algorithm 1. The particularity of GGGP methods is to ensure individuals are still consistent with grammar rules after each crossover and mutation operation. In order to select the parents of the next generation, the performance of each individual in the current generation is assessed using an optimization objective (called fitness function). The algorithm stops either when reaching the maximum number of iterations or when a tree-program has a higher score than a satisfaction threshold.

To constrain the search space, we asked operators to look at the variables created by the algorithm, to extract relevant characteristics from them and to propose new grammatical rules. These iterations allowed us to create the grammar described above.

4.3 Objective Function

We use the absolute value of the correlation coefficient r as a measure of the efficiency of each individual. This measure, based on the linear Pearson correlation, lies between 0 and 1, where a r value of 1 indicates that the predictions $yhat$ perfectly matches the behavior of the target y. While the linear correlation coefficient mainly measures the strength of the linear relationship between two variables, it also has a clear advantage to be able to compare the behavior of two variables which range on different scales.

This measure seems particularly well suited in this particular case because we are not interested here in predicting the exact flow value but rather to understand the *global underlying relationship* between input variables X and flow output y.

Algorithm 1: Interactive Evolutionary Search Algorithm

```
input : observations X, target y
grammar ← initialize_grammar()
while operator_not_satisfied do
    population ← create_population(grammar)
    evolutionary_search_condition_not_met ← True
    while evolutionary_search_condition_not_met do
        parents, best_individual ← parents_selection(population, X, y)
        offspring ← crossover(parents)
        offspring ← mutation(offspring)
        population ← replacement(population, offspring)
        evolutionary_search_condition_not_met ← test_conditions(best_individual)
    grammar ← operator_grammar_update(grammar)
return grammar, best_individual
```

Moreover, from an operator viewpoint, it is as interesting to look at one feature F as its rescaled value $10 \times F$. To understand the advantages of using this metric, we also compare it to the distance-based metric Mean Squared Error (MSE).

5 Experiments

5.1 Target Selection

As we are only interested in analyzing flows y on *sensitive* lines, we first selected a subset of all 92 lines on which we'll perform feature extraction. This preprocessing step ensures that we won't look at residual information or negligible effects on the target. To do so, we identified which are the *most frequently loaded lines* by selecting the ones above a designated percentage *percent_i_threshold* (100, 90, 80, and 70%) of the line thermal limit *i_threshold* during a percentage of total time *percent_time* (0.05 or 0.1%). The final target selection is defined as the union of lines identified as loaded by one of each combination of hyperparameters (*percent_i_threshold*, *percent_time*). Using this method, we detected 24 lines.

5.2 Settings

Experimental Protocol For each line identified as sensitive, with a corresponding target flow y, we now search for a tree-like variable to represent it. We evolve a large population of 2 000 individuals over 200 generations by following the grammatical rules defined above. A large population is proved necessary due to the high number of constraints defined in the grammar. The initial dataset containing 365 165 observations is split following a 80/20% ratio between the train and test sets. Each line feature search is launched 30 times with random population initialization, and only the top features are kept for manual inspection. All hyperparameters are provided in Table 1 for reproducibility, and were selected during preliminary cross-validation experiments.

Parameter	Setting
Generation	200
Population size	2000
Initialization	PI Grow with max inital depth 10
Selection	Tournament with a size of 2
Crossover	Type variable one point (0.9 probability)
Mutation	Type int flip per codon (0.1 probability)
Elitisme	Top 10
Replacement	Generational
Fitness	Absolute Pearson correlation or MSE

Table 1. Evolutionary parameters chosen for the experiment.

For the two fitness metrics (correlation and MSE) in both experiments, we insert an additive regularization term, relative to the depth of the feature-tree: `individual_depth`. The depth is the maximum number of nodes in the feature from the root to any leaf. This constraint aims at preventing the tree size explosion (called bloating phenomenon) [37] and is slightly weighted to only remove redundant nodes without constraining the search-space too much. Eventually our fitness function is: `fitness = selected_metric +` 10^{-8} `* individual_depth`

Implementation To take advantage of their parallelized implementation, we used the open-source implementation of GE in Python PonyGE2 [11] as backbone code. We inserted correlation-based error-metrics, a specialized data processing, custom evolutionary step, and the full grammar tailored to our problem.

5.3 Results

Experiment 1 : Metrics Comparison In this first experiment, we conduct two parallel trials, where we only vary the fitness metric used to evolve and assess the top-individuals performance line per line. Our objective is to identify the adequate fitness to our problem, choosing between distance-based methods such as Mean Squared Error (MSE) and correlation-based methods such as absolute Pearson correlation. Obtained results on the test set are summarized in Figures 2 and 3.

Figure 2 details the results of the two trials (MSE or Pearson as fitness) for 30 runs and compares them using MSE metric on a log-scale. In this Figure, two boxplots are associated to each line. They are placed on each side of a vertical dotted line: on the left the blue boxplot corresponds to the top-individuals evolved with MSE fitness ; on the right an orange boxplot contains scores of correlation-based evolutions. Similarly, Figure 3 compares the same two trials regarding the absolute value of the Pearson correlation of the top-individual.

By comparing power line per power line the correlation-based and distance-based evolutions using correlation (Figure 3) and MSE (Figure 2) metrics, we identify that the two evolutions produce contrastive individuals, with top MSE-fitness individuals usually performing far worse than correlation-fitness individuals

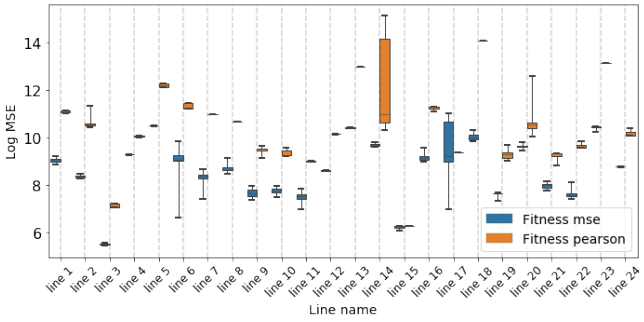

Fig. 2. Comparison of two evolution strategies, either using MSE (in blue) or Pearson Correlation (in orange) as fitness on 30 runs. Each boxplot summarizes the log-MSE scores for each line's best individuals (i.e., for each run, the one with the highest fitness score at the end of the evolution). Figures are best seen in color.

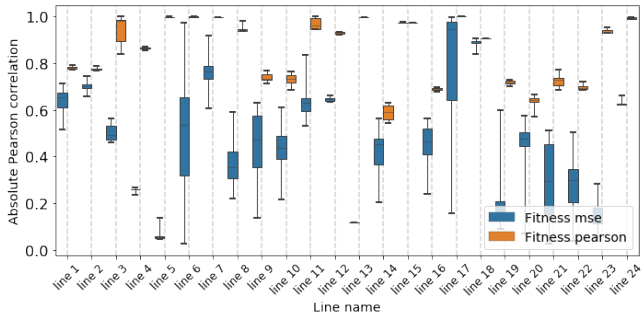

Fig. 3. Comparison of two evolution strategies on the same 30 runs presented in Figure 2. Each boxplot now represents the absolute Pearson Correlation score for the best individuals of each line. Individuals compared in Figures 2 and 3 are identical.

on the Pearson correlation scale (and conversely on the log-MSE scale). Except for few lines, where both fitness metrics performed well (ex. lines 3 and 15 on a log-MSE scale, 15 and 17 on correlation scale which have close scores whatever the evolution fitness metric), the 2 metrics seem to explore the feature space in opposite directions and can't be used equivalently. Moreover, when analyzing the features produced using MSE fitness, we identify that they tend to select combinations of features preferentially from the initial dataset with value ranges similar to the target one (although their behavior is different). Unlike individuals created with the correlation fitness, features obtained with MSE fitness wouldn't have a physical/technical interest and would mainly use the feature with intensity variables i. This point is critical to use distance-based fitness metrics because we can't normalize our data to have similar ranges as it would result in the impossibility to respect physical laws. For example, after normalizing each input features independantly, Kirchhoff's circuit laws don't apply anymore. Based

on these observations, the only acceptable strategy is then to use fitness-based metrics such as Pearson correlation.

Regarding the outputted features, this first experiment also identified that we couldn't have constructed fewer features than the number of lines without loosing in performance, because extracted variables are very different from one line to another.

Experiment 2: Comparison to Other Methods In the second experiment, we compare individuals from experiment 1 (obtained through evolution with a correlation-based fitness) with the output of algorithms such as LASSO Lars with Bayes Information criterion (Lasso Lars-BIC) [51], depth-3 Decision Tree [5]. These algorithms were selected because of their capacity to give outputs with a comparable level of interpretability, thus falling under the "Functionally-grounded" evaluation [9]. We also show the most correlated feature from the original dataset as a baseline. The obtained results are presented in Figure 4. In this figure, the results associated with each line are displayed along a dotted vertical line and labeled at the bottom by their corresponding line name. Thus, for line 24 on the right, we have from the bottom to the top: the most correlated feature from the initial dataset (marked by a red star); the correlation to Lasso Lars-AIC output (pink rectangle); the correlation to a Decision Tree output (blue diamond) and the boxplot of our GGGP method (orange box).

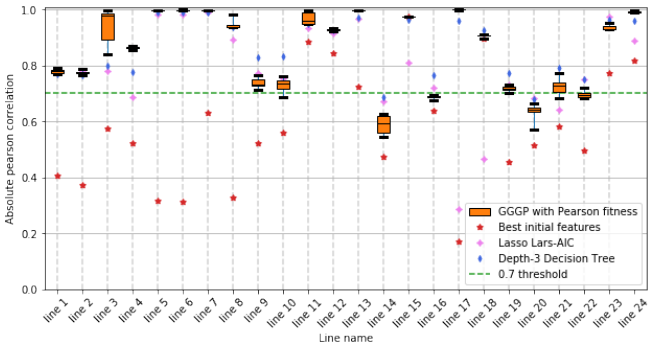

Fig. 4. Comparison between GGGP and state-of-the-art-algorithms.

As presented in Figure 4, all GGGP outputs have a higher correlation to the target than features from the initial dataset. Furthermore, by looking at only at boxplots, we identify that only 3 out of 24 highest values in boxplots were under the 0.7 threshold, under which correlation is usually not considered strong enough for the feature to be significant. We finally compare our approach in terms of correlation with partially interpretable ones such as Lasso Lars or a depth-constrained Decision Tree. From these experiments, we can highlight that GGGP outputs with significant correlation are at least as highly and sometimes

even more correlated to the target than outputs from other methods. In the next paragraph, we'll look in detail at the produced combination categories.

Human-expert Output Analysis and Pieces of Advice Eventually, the relevance of top features obtained with GGGP-method is technically assessed by power system experts to identify whether the obtained formulas are conclusive from a technical and physical perspective, could be useful to operators, and would make sense to them. This experiment aims at confirming outputs interpretability from a human perspective by performing a "Human-Grounded" evaluation [9]. The first conclusions are that all features above a 0.8 correlation are relevant even if some of them could be improved. Thus, 0.8 could then be used as an acceptation threshold below which features created would be rejected. Indeed, extracted features with a very high score (above 0.9) show a small discrepancy between runs and could be useful as-is. However, for features under 0.8, experts would have found it interesting to intervene during the learning process by removing, replacing, or adding nodes or leaves in the evolved trees to increase their score. Uncovering the literal expression of features, we identify groups of features with intriguing expert interpretations :

- some features are variations around the expression $\sqrt{p^2+q^2}$ (such as $\frac{\sqrt{p^2+q^2}}{v}$ or $\sqrt{(p1+p2)^2+(q1+q2)^2}$)
- some others are the sum, minimum or maximum of a list of active powers, tweaked using absolute value, positive or negative parts. In some cases, these non-linearities are also found useful to tackle outliers coming from sensor or simulation errors.
- few features are also an aggregation (sum or difference) of cross-border flows (ex. $i\_frontier_1 - abs(i\_frontier_2)$). These combinations would tell us that the corresponding target line is more sensitive to a *global* phenomenon than other lines. It could allow to identify which lines are sensitive to a high flow coming inside or leaving the zone.
- using a specific grammar version, some of those features could even include multiple features combined using graph topology as conditions, such as :

```
if{line_1 is connected} then{monitor gggp_feature_1}
else{monitor gggp_feature_2}
```

6 Conclusion and Future Works

We have shown that the proposed interactive GGGP method achieved promising results on *interpretable feature extraction*. We marked the first milestone with first the production and then both qualitative and quantitative validation of a custom context-free grammar, which could interactively include some power system knowledge. Our experiments also provide some insights on the interpretability of our method from "Human-Grounded" and "Functionally-Grounded" [9] perspectives. By introducing expertise and physical properties in the grammar rules, we obtained explainable features. Some of them were also found relevant

enough by power system operators to be included in hyper-vision tools. However, a few target features were still tricky to handle with only one dimension. Indeed, for these few lines, a 1D-manifold is surely too restrictive, and we envision that building a multi-dimensional space could highly increase the representativity of the reduced space. We also identified the use of Probabilistic Grammars to enhance more precise space exploration.

Moreover, as these new features are made to be used by operators, the very next step will be to introduce interactivity with non-machine-learning experts, directly inside evolutionary runs [1]. We would allow them to provide insights and technical information that could significantly help to create more insightful representation: either selecting/removing individuals, inforcing constraints on the search space by iteratively changing the grammar over the generations.

Undergoing works also focus now on applying the same method to a wider geographical zone only on high voltage power lines (low voltage power lines modeled as aggregated consumptions). This perspective brings us closer to performing an "Application-Grounded" evaluation of our method with humans on a more complex task. Eventually, to release reproducible results on power systems test cases and open-source our code, we plan to use the open-source framework Grid2Op[1] developed to test machine learning strategies for power grid operations.

References

1. Amershi, S., Cakmak, M., Knox, W.B., Kulesza, T.: Power to the people: The role of humans in interactive machine learning. Ai Magazine **35**(4), 105–120 (2014)
2. Banzhaf, W., Francone, F.D., Keller, R.E., Nordin, P.: Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications. Morgan Kaufmann Publishers Inc. (1998)
3. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation **15**(6), 1373–1396 (2003)
4. Bohanec, M., Bratko, I.: Trading accuracy for simplicity in decision trees. Machine Learning **15**(3), 223–250 (1994)
5. Breiman, L., Friedman, J.H., Olshen, R.A., et al.: Classification and Regression Trees. Wadsworth (1984)
6. Charrier, C., Lezoray, O.: Color vq-based image compression by manifold learning. In: ICISP. vol. 6134, pp. 79–85. Springer (2010)
7. Cherrier, N., Poli, J., Defurne, M., Sabatié, F.: Consistent feature construction with constrained genetic programming for experimental physics. In: IEEE Congress on Evolutionary Computation, CEC. pp. 1650–1658 (2019)
8. Donnot, B., Guyon, I., Schoenauer, M., Panciatici, P., Marot, A.: Introducing machine learning for power system operation support. In: IREP Symposium (2017)
9. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. stat **1050**, 2 (2017)
10. Evans, B.P., Xue, B., Zhang, M.: What's inside the black-box?: a genetic programming method for interpreting complex machine learning models. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 1012–1020 (2019)

[1] Grid2Op github repository : `https://github.com/rte-france/Grid2Op`

11. Fenton, M., McDermott, J., Fagan, D., et al.: Ponyge2: grammatical evolution in python. In: GECCO (Companion). pp. 1194–1201 (2017)
12. Friedman, J.H., Popescu, B.E., et al.: Predictive learning via rule ensembles. The Annals of Applied Statistics **2**(3), 916–954 (2008)
13. Hastie, T.J.: Generalized additive models. Wiley Online Library (2017)
14. Hein, D., Udluft, S., Runkler, T.A.: Interpretable policies for reinforcement learning by genetic programming. Engineering Applications of Artificial Intelligence **76**, 158–169 (2018)
15. Hosseini, B., Hammer, B.: Interpretable discriminative dimensionality reduction and feature selection on the manifold. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 310–326. Springer (2019)
16. Jolliffe, I.T.: Principal Component Analysis. Springer Series in Statistics, Springer Verlag (1986)
17. Khalid, S., Khalil, T., Nasreen, S.: A survey of feature selection and feature extraction techniques in machine learning. In: 2014 Science and Information Conference. pp. 372–378. IEEE (2014)
18. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
19. Knuth, D.E.: backus normal form vs. backus naur form. Commun. ACM **7**(12), 735–736 (1964)
20. Koza, J.R.: Concept formation and decision tree induction using the genetic programming paradigm. In: Parallel Problem Solving from Nature, 1st Workshop, PPSN I, Dortmund, Germany, Proceedings. vol. 496, pp. 124–128 (1990)
21. Koza, J.R.: Hierarchical automatic function definition in genetic programming. In: Proceedings of the Second Workshop on Foundations of Genetic Algorithms. pp. 297–318. Morgan Kaufmann (1992)
22. Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G.: Genetic programming IV: Routine human-competitive machine intelligence, vol. 5. Springer Science & Business Media (2006)
23. Laugel, T., Lesot, M., Marsala, C., et al.: The dangers of post-hoc interpretability: Unjustified counterfactual explanations. In: IJCAI. pp. 2801–2807 (2019)
24. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Advances in neural information processing systems. pp. 556–562 (2001)
25. Lensen, A., Xue, B., Zhang, M.: Can genetic programming do manifold learning too? In: Genetic Programming - 22nd European Conference, EuroGP 2019, Held as Part of EvoStar 2019, Proceedings. vol. 11451, pp. 114–130. Springer (2019)
26. Lensen, A., Xue, B., Zhang, M.: Genetic programming for evolving a front of interpretable models for data visualization. IEEE Transactions on Cybernetics (2020)
27. Li, J., Cheng, K., Wang, S., et al.: Feature selection: A data perspective. ACM Computing Surveys (CSUR) **50**(6), 1–45 (2017)
28. Li, Y., Pan, Q., Wang, S., et al.: Disentangled variational auto-encoder for semi-supervised learning. Information Sciences **482**, 73–85 (05 2019)
29. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems 30, pp. 4765–4774. Curran Associates, Inc. (2017)
30. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(Nov), 2579–2605 (2008)
31. Marot, A., Rosin, A., Crochepierre, L., Donnot, B., Pinson, P., Boudjeloud-Assala, L.: Interpreting atypical conditions in systems with deep conditional autoencoders: The case of electrical consumption. In: ECML/PKDD (3). Lecture Notes in Computer Science, vol. 11908, pp. 638–654. Springer (2019)

32. Marot, A., Tazi, S., Donnot, B., Panciatici, P.: Guided machine learning for power grid segmentation. In: 2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe). pp. 1–6. IEEE (2018)
33. McInnes, L., Healy, J., Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426 (2018)
34. McKay, R.I., Hoai, N.X., Whigham, P.A., Shan, Y., O'Neill, M.: Grammar-based genetic programming: a survey. Genetic Programming and Evolvable Machines **11**(3-4), 365–396 (2010)
35. Neshatian, K., Zhang, M., Andreae, P.: A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming. IEEE Transactions on Evolutionary Computation **16**(5), 645–661 (Oct 2012)
36. Polito, M., Perona, P.: Grouping and dimensionality reduction by locally linear embedding. In: NIPS. pp. 1255–1262. MIT Press (2001)
37. Purohit, A., Bhardwaj, A., Tiwari, A., Chaudhari, N.S.: Handling the problem of code bloating to enhance the performance of classifier designed using genetic programming. In: IICAI. pp. 333–342 (2011)
38. Raducanu, B., Dornaika, F.: A supervised non-linear dimensionality reduction approach for manifold learning. Pattern Recognition **45**(6), 2432–2444 (2012)
39. Ratle, A., Sebag, M.: Genetic programming and domain knowledge: Beyond the limitations of grammar-guided machine discovery. In: Parallel Problem Solving from Nature-PPSN VI, 6th International Conference. vol. 1917, pp. 211–220 (2000)
40. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should I trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1135–1144 (2016)
41. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence **1**(5), 206–215 (2019)
42. Ryan, C., Collins, J.J., O'Neill, M.: Grammatical evolution: Evolving programs for an arbitrary language. In: Genetic Programming, First European Workshop, EuroGP'98, Proceedings. vol. 1391, pp. 83–96. Springer (1998)
43. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation **10**(5), 1299–1319 (1998)
44. Sereeter, B., Vuik, C., Witteveen, C.: On a comparison of newton–raphson solvers for power flow problems. Journal of Computational and Applied Mathematics **360**, 157–169 (2019)
45. Sohn, K., Lee, H., Yan, X.: Learning structured output representation using deep conditional generative models. In: NIPS. pp. 3483–3491 (2015)
46. Tenenbaum, J.B., De Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. science **290**(5500), 2319–2323 (2000)
47. Van Der Maaten, L., Postma, E., Van den Herik, J.: Dimensionality reduction: a comparative. J Mach Learn Res **10**(66-71), 13 (2009)
48. Vilenchik, D., Yichye, B., Abutbul, M.: To interpret or not to interpret pca? this is our question. In: ICWSM. pp. 655–658. AAAI Press (2019)
49. Vlachos, M., Domeniconi, C., Gunopulos, D., et al.: Non-linear dimensionality reduction techniques for classification and visualization. In: KDD. pp. 645–651 (2002)
50. Wu, C., Ioannidis, S., Mario, S., et al.: Iterative spectral method for alternative clustering. In: Artificial Intelligence and Statistics (2018)
51. Zou, H., Hastie, T., Tibshirani, R., et al.: On the "degrees of freedom" of the lasso. The Annals of Statistics **35**(5), 2173–2192 (2007)