Radar Emitter Signal Recognition Based on Automated Machine Learning Workflow Optimization (Postprint)

Authors: Tu Tongheng, Jin Weidong

Date: 2018-05-20T00:00:00+00:00

Abstract

To address the complex parameter configuration issues in radar emitter signal recognition, this study investigates machine learning parameter optimization and proposes a tree-structure-based machine learning pipeline optimization method. This method utilizes genetic programming to generate tree-structure-based machine learning pipelines and evolves both their structure and parameters to obtain the optimal parameterized machine learning pipeline. This pipeline can include arbitrary combinations of feature processing and modeling, enabling learning and recognition from raw datasets. Compared with manually parameterized one-vs-one Support Vector Machines on two radar signal feature sets of different dimensions, this method eliminates the need for tedious parameter configuration and achieves a maximum accuracy improvement of over 6%, demonstrating that the tree-structure-based machine learning pipeline obtained by this method possesses significant advantages.

Full Text

Radar Emitter Signal Recognition Based on Optimization of Automatic Machine Learning Pipeline

Tu Tongheng, Jin Weidong
School of Electrical Engineering, Southwest Jiaotong University, Chengdu 610031, China

Abstract

To address the complex parameter configuration problem in radar emitter signal recognition, this paper investigates machine learning parameter optimization and proposes a tree-based machine learning pipeline optimization method.

This approach employs genetic programming to generate tree-structured machine learning pipelines and evolves both their structure and parameters to obtain the best-performing parameterized pipeline. The pipeline can include any combination of feature processing and modeling operations to learn from and recognize original datasets. Compared with manually configured one-vs-one support vector machines on two radar signal feature sets of different dimensions, the proposed method eliminates cumbersome parameter tuning while achieving accuracy improvements of over 6%, demonstrating clear advantages of the tree-based machine learning pipeline.

Keywords: auto ML; hyperparameter optimization; genetic programming; radar emitter signal; SVM

0 Introduction

Modern battlefield electromagnetic environments present increasing challenges for radar emitter signal recognition due to the deployment of new radar systems such as phased array radar and the dramatic increase in the number of emitters. Signals have become denser and more complex in pattern, demanding higher recognition capabilities. Current research primarily focuses on three aspects: feature extraction, feature selection, and classifier design, with machine learning being the dominant methodology. Commonly used signal features include directly measured parameters such as Time of Arrival (TOA), Radio Frequency (RF), Pulse Width (PW), Pulse Amplitude (PA), and Direction of Arrival (DOA), as well as increasingly emphasized intra-pulse characteristics. Primary machine learning methods employed include Support Vector Machine (SVM), Artificial Neural Network (ANN), and Random Forest (RF).

Radar emitter signal recognition fundamentally constitutes a machine learning problem. A typical machine learning workflow involves data cleaning, feature extraction, feature construction, feature selection, and finally classification using a parameter-optimized model. Throughout this pipeline, researchers must transform data to make it more suitable for modeling—for instance, through feature normalization (feature transformation), removal of less useful features (feature selection), or creation of new features from existing data (feature construction). Subsequently, they must select appropriate machine learning models and optimize their parameters to ensure final classification accuracy. Even experienced experts struggle to configure these steps easily, let alone other personnel facing increasingly complex signals.

Over the past two decades, intelligent systems have advanced dramatically, demonstrating human-surpassing performance in diverse tasks such as space antenna design, vulnerability detection in large software projects, and playing Go. This raises a compelling question: Can intelligent systems automatically design machine learning pipelines? The answer is affirmative. Current research on machine learning pipeline optimization includes methods such as Bayesian optimization, grid search, and bilevel optimization. While automatic machine

learning (auto ML) has traditionally focused on model parameter optimization to maximize classification accuracy, recent studies show that random search can be more effective than exhaustive search for discovering ideal parameters, with Bayesian optimization particularly yielding results that often surpass manual tuning.

This paper employs the Tree-based Pipeline Optimization Tool (TPOT), which uses Genetic Programming (GP) to automatically design and optimize machine learning pipelines. By integrating the characteristics of radar emitter signal recognition with the advantages of pipeline optimization, we introduce this tool to the field and configure genetic programming settings to maximize classification accuracy while reducing pipeline complexity.

1 Methodology

1.1 Operators in the Pipeline

1.1.1 Feature Preprocessing TPOT utilizes three preprocessing methods: Standard Scaler, Robust Scaler, and Polynomial Features. Standard Scaler scales feature weights based on sample mean and variance, while Robust Scaler employs median and interquartile range for robust feature scaling. Polynomial Features generates interaction features through polynomial combinations of numerical features.

1.1.2 Decomposition For dataset decomposition and dimensionality reduction, TPOT uses Randomized PCA, a variant of principal component analysis employing randomized singular value decomposition.

1.1.3 Feature Selection Feature selection methods include RFE, Select KBest, Select Percentile, and Select Percentile (note: the original text lists four methods but names only three distinct strategies). These employ recursive feature elimination, selection of top-k features, selection of top-n percentile features, and removal of features below minimum variance thresholds, respectively.

1.1.4 Model Selection Since recognition targets are primarily labeled data samples, TPOT uses supervised learning models including tree-based classifiers (decision tree, random forest, gradient boosting) as well as SVM, logistic regression, and K-Nearest Neighbors (KNN). All operators are implemented using the scikit-learn machine learning library, a general-purpose Python machine learning library.

1.2 Pipeline Construction

Conventional machine learning requires humans to build features, select features, choose appropriate classifiers, and set optimal parameters based on extensive experimental experience. Different recognition objects demand different

pipeline configurations, posing significant challenges for manual tuning. This paper employs a tree-based automatic machine learning pipeline designed through genetic programming for optimization. [Figure 1: see original paper] illustrates the structure of a typical optimized pipeline, where boxes represent operators. The diagram shows the general structure without specifying particular operations. As data flows through these operators, features are added, removed, or modified through preprocessing and principal component analysis, followed by classification model selection with corresponding parameters.

The four operator categories mentioned above serve as nodes in the genetic programming tree structure. An additional operator—feature merging—combines multiple inputs into a single input. All GP-generated pipelines begin with the original dataset and its copies (the tree leaves), followed by random sequences of the four operator categories. The specific operators and their applications are also randomly determined. After passing through feature preprocessing, decomposition, and feature selection operations, the output remains a feature set that can continue through additional nodes. When multiple inputs precede a node, the feature merging operator combines them into a single feature set for downstream processing.

Data passing through model selection nodes produces classification predictions. If not the final modeling node, these predictions are added as new features to the input feature set, with previous predictions removed at each new modeling node. During evolution, both model parameters and other operation parameters are subject to optimization. The final node must be either a modeling node or a merge of multiple modeling nodes, with its prediction results serving as the metric for evaluating overall pipeline performance. This tree-based structure allows arbitrary variation in node quantity and relationships, enabling pipelines of any configuration. We split data into 75% training and 25% testing sets, with all pipelines trained exclusively on the training set and evaluated on the test set.

1.3 Genetic Programming

Genetic programming draws inspiration from natural biological evolution and genetic theory, representing an automated method for randomly generating search programs. As a novel global optimization algorithm characterized by simplicity, universality, and robustness, GP has demonstrated strong problem-solving capabilities for complex nonlinear problems and has been successfully applied across many domains with increasing research depth in recent years.

To automatically generate the tree-based pipelines described above, we employ genetic programming implemented using the DEAP Python library. In this context, GP constructs tree structures with pipeline operators as nodes to maximize final classification accuracy. The algorithm evolves both the operational flow acting on datasets and node parameters, such as the number of trees in a random forest or the number of features selected during feature selection.

The GP algorithm follows standard evolutionary procedures with parameters listed in . Each evolution run begins with randomly generating a fixed number of tree-structured pipelines to form the initial population. These pipelines are evaluated based on classification accuracy, which serves as their fitness measure.

Table 1: Genetic Programming Parameter Settings

GP Parameter	Value
Individual mutation rate	(value not specified in text)
Individual crossover rate	(value not specified in text)
Selection method	3-tournament selection eliminating lowest fitness, then choosing lower complexity from remaining two
Replacement strategy	10% elite retention, complexity-based 2-choice-1 replacement
Mutation operators	Insert, delete, replace, each accounting for 1/3 of mutations

After evaluating all pipeline individuals, the system proceeds to next-generation creation. To produce offspring, it first copies the highest-fitness individuals into the child population until these elites constitute 10% of the population (elite retention strategy). The remaining 90% is filled through tournament selection: randomly selecting three individuals, eliminating the lowest-fitness one, then choosing the less complex (fewer operation nodes) from the remaining two for replication into the next generation.

Following selection, a point crossover operator acts on the copied individuals selected by crossover rate, randomly choosing two individuals and swapping their contents at a random pipeline structure point. Individuals selected by mutation rate undergo three mutation types: replacement (randomly substituting an operation node with a new random sequence), insertion (adding a new random sequence at a random position), and deletion (removing a random sequence segment). Each mutation operator has a 1/3 probability of acting on a selected individual. All crossover and mutation operations prohibit invalid pipelines, such as those passing datasets to nodes expecting single parameters.

After completing crossover and mutation, the previous generation is entirely discarded, and the evaluation-selection-crossover-mutation process repeats for a fixed number of generations. Through this mechanism, GP continuously modifies pipelines by adding beneficial operations and removing redundant or

detrimental ones. During evolution, the system tracks and stores the single best-performing pipeline separately, which becomes the final optimization result upon completion.

2 Radar Emitter Signal Feature Sets

Due to project involvement, this study focuses on radar emitter signals obtained through computer simulation and provided by an electronics research institute in Southwest China. The dataset comprises two types: radar pulse interception fragments and pulse time series with associated features.

The feature set includes the following radar emitter signal parameters and characteristics: Time of Arrival (TOA), Radio Frequency (RF), Pulse Width (PW), Pulse Amplitude (PA), Differential Time of Arrival (DTOA), intra-pulse start position, intra-pulse end position, and modulation type (radar ID). From this set, we select RF, PW, and PA as the original dataset features, with modulation type serving as sample labels.

Additionally, we extract intra-pulse features including Information Fractal dimension (IF), Information Entropy (IE), Lemple-Ziv (LZ) complexity, and Wavelet Ridge Frequency features. Intra-pulse characteristics primarily manifest in frequency, phase, and amplitude variations. The presence and type of intra-pulse modulation directly reflect in signal waveforms, enabling recognition through complexity metrics—IF, IE, and LZ all belong to complexity features. Wavelet transform offers adaptive resolution that automatically adjusts to signal characteristics, making it highly suitable for analyzing non-stationary signals. This "mathematical microscope" property allows examination of both overall signal structure and detailed features, facilitating radar signal modulation type recognition through wavelet transform characteristics.

All intra-pulse features are extracted from pulse time series using MATLAB tools and combined with RF, PW, and PA to form an alternative feature set for subsequent experiments. Samples from both feature sets are shuffled, with 1,000 samples randomly selected from each set. These samples contain 100 instances from each of ten radar modulation types, constituting the final feature sets after format adjustment for the optimization tool.

3 Experimental Results and Analysis

In addition to the best-performing machine learning pipeline generated by TPOT using genetic programming, we introduce conventional one-vs-one SVM as a comparative baseline. Experiments are conducted on two feature sets: Feature Set 1 comprising RF, PW, and PA; and Feature Set 2 incorporating intra-pulse features. All experiments split data into 75% training and 25% testing sets.

[Figure 2: see original paper] and [Figure 3: see original paper] illustrate classification performance of SVM and TPOT on both feature sets. Evidently, the

tree-structured machine learning pipeline obtained through genetic programming demonstrates superior performance. On average, classification accuracy exceeds SVM by at least 6 percentage points on both feature sets. The results also reveal more stable performance compared to SVM' s significant fluctuations. Furthermore, incorporating intra-pulse features effectively improves accuracy, with both optimized pipelines and SVM gaining at least 1 percentage point when intra-pulse features are added.

[Figure 4: see original paper] displays the tree-structured machine learning pipeline optimized on Feature Set 2. In this pipeline, the original dataset is copied into three branches for parallel feature processing and classification, with results merged for final classification. The optimization includes specific parameters (omitted here for brevity). While not a simple pipeline and requiring considerable evolution time on large datasets, the final classification execution remains efficient. However, incorporating Artificial Neural Networks (ANN) would likely increase evolution time substantially, presenting a significant challenge for future development toward intelligent learning directions.

4 Conclusion

This paper identifies an alternative automatic machine learning approach from parameter optimization research: automatically generating tree-structured machine learning pipelines and optimizing their structure and parameters through genetic programming to obtain optimal classification pipelines within the search space.

Experiments on radar emitter signal feature sets demonstrate that this method achieves excellent and stable classification performance. However, genetic operations may require substantial time for large datasets. In our experiments, while maximum computation time reached 12 hours, this remains more efficient and meaningful in the long term than manual repetitive experiments yielding suboptimal parameters.

The experiments also show that combining effective features enables better automatic learning pipeline performance. Future research could explore introducing more effective features or extraction algorithms. Currently, the method does not incorporate AI algorithms such as ANN; future work could attempt pipeline and parameter optimization for these algorithms, though this would demand even more substantial computation time. Therefore, reducing evolution time while maintaining classification effectiveness represents a key focus for future research.

References

[1] Chen Changxiao, He Minghao, Xu Jing, et al. Research progress on radar emitter recognition technology [J]. Journal of Air Force Early Warning Academy, 2014, 28(1): 1-5.

[2] Zhang Gexiang. Research on intelligent recognition methods for radar emitter signals [D]. Chengdu: Southwest Jiaotong University, 2005.

[3] Yan Youbiao, Chen Yuanyan. Overview of main strategies in machine learning [J]. Application Research of Computers, 2004, 21(7): 4-10.

[4] Olson R S, Bartley N, Urbanowicz R J, et al. Evaluation of a tree-based pipeline optimization tool for automating data science [C]// Proc of Genetic and Evolutionary Computation Conference. 2016: 485-492.

[5] Olson R S, Moore J H. TPOT: A tree-based pipeline optimization tool for automating machine learning [J]. Journal of Machine Learning Research, 2016, 64: 66-74.

[6] Klein A, Falkner S, Bartels S, et al. Fast Bayesian optimization of machine learning hyperparameters on large datasets [J/OL]. 2016. https://arxiv.org/abs/1605.07079.

[7] Li Shaobo, Hu Jianjun. Genetic Programming and Innovative Design of Electromechanical Systems [M]. Beijing: Mechanical Industry Press, 2009: 61-62.

[8] Pedregosa F, Varoquaux G, Gramfort A, Michel V, et al. Scikit-learn: machine learning in python [J]. J Mach. Learn. Res, 2011, 12: 2825-2830.

[9] Zha Zhiqin, Gao Bo, Zheng Chengzeng. Research on genetic programming implementation [J]. Computer Applications, 2003, 23(7): 137-139.

[10] Fortin, F. A, Gardner, M. A, Parizeau, M, Gagne, C, et al. DEAP: evolutionary algorithms made easy [J]. J Mach. Learn. Res, 2012, 13: 2171-2175.

[11] Yu Zhibin. Research on radar emitter signal recognition based on intrapulse features [D]. Chengdu: Southwest Jiaotong University, 2010.

Note: Figure translations are in progress. See original paper for figures.