Prototype-pairs Decomposition for Extracting Simple and Meaningful Rules

Marcin Blachnik[1 [0000−0003−3336−4962]],
Mirosław Kordos[2 [0000−0002−2031−7561]], and
Daniel Dąbrowski[1 [0009−0007−4250−7060]]

Silesian University of Technology,
Department of Industrial Informatics,
40-019 Katowice ul. Krasińskiego 8, Poland
{marcin.blachnik, daniel.dabrowski}@polsl.pl
University of Bielsko-Biała,
Department of Computer Science and Automatics,
43-309 Bielsko-Biała ul. Willowa 2, Poland
mkordos@ubb.edu.pl

Abstract. We present a preliminary study of a model-agnostic method called prototype pair decomposition that generates simple and accurate decision rules from datasets. The research focuses on its application to decision trees. It starts by selecting representative prototypes obtained by a prototype construction method, then pairs of prototypes from opposite classes are determined. These pairs define subspaces containing a fragment of the decision boundary in which a shallow decision tree is applied to extract simple decision rules consisting of a few premises. The results indicate that the proposed solution allows the extraction of locally competent simple rules that are comparable in terms of classification accuracy to a large and complex set of global rules obtained from standard decision trees.

Keywords: Explainable AI · Decision trees · Prototype-Based Learning

1 Introduction

The explanation of data and the decisions taken by machine learning models have been investigated since the beginning of AI research [12]. The problem is constantly gaining importance as the data get bigger and more complex. It is important that the user not only knows the prediction result, but also understands why a given result was reached by the model. This allows users to trust the models, which in many cases is a condition to apply them in practice [10].

Machine learning methods are often categorized as black boxes or white (glass) boxes. Black box models, like kernel methods, neural networks, and ensembles (e.g., random forest), typically offer high accuracy but limited interpretability. In contrast, white box models—such as decision trees, sequential covering, case-based reasoning (CBR) [4], and prototype-based rules [2]—provide interpretable decisions, often at the cost of accuracy.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_11

To interpret black-box models, two main approaches are used [11]: model-independent methods and those tailored to specific models. Model-independent methods extract rule-based explanations by labeling data with predictions from black-box models. For instance, [7] approximates neural network activations with linear fragments to extract rules. Knowledge extraction from deep networks using CBR is shown in [8]. While kernel-based models interpretation as prototype-based rules can be found in [1].

Another strategy is local interpretation, as in LIME [10], which fits a linear model near the query point—later extended with autoencoders [13]. Graphical explanations like SHAP [6] and saliency maps [14] offer local insights, particularly for convolutional networks.

In this paper, we propose for the first time a preliminary study on a model-agnostic method for rule extraction that leverages problem decomposition through a technique called prototype pair decomposition (PPD) to improve the incorporability of decision boundaries in classification problems. The proposed algorithm integrates case-based reasoning with classical crisp rules by constructing locally competent and interpretable rules. This is achieved by identifying a pair of adversarial (enemy) prototypes, which are prototypes of opposite classes. This pair partitions the decision boundary of the classification problem into regions by assigning each training sample to it nearest pair of prototypes. Since the prototypes belong to different classes, the resulting pair defines a fragment of the decision boundary that is locally approximated by a simple set of rules. From a system-wide perspective, the proposed solution decomposes the classification problem into regions formed by subsets of the original dataset. Each region is identified by a pair of prototypes, and in each region, an independent model can be trained.

In our study we use a decision tree as a reference model for measuring the benefits of the proposed solution. We show that the rules obtained from decision trees with the PPD algorithm are much simpler that those generated by standard decision trees trained on the entire training data.

2 The prototype-pairs decomposition algorithm

The PPD algorithm (see sketch 1) requires a training set $\mathbf{T} = \{\mathbf{X}, \mathbf{y}\}$ and the number of prototypes k as input. It is assumed that $y_i \in \{P, N\}$, that is a binary classification problem.

PPD starts by selecting representative prototypes obtained by any instance selection or prototype construction method [3]. We use a clustering-based prototype selection, clustering each class separately. Then the set of possible pairs $\mathbf{U}$ is determined by identifying the nearest positive and negative prototype for each training sample (adversarial prototypes). Thus, each pair consists of one instance from a positive and one from a negative class. The regions are encoded using the Cantor pairing function which allows encoding a pair by a single integer value. Each pair defines the decision boundary of the nearest neighbor classifier. The pairs define subspaces also called regions, which are the key concept of the

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_11

Algorithm 1 A PPD algorithm

function PPD(**X**,**y**,k)
  $(\mathbf{P}_p, \mathbf{P}_n) \leftarrow$ FindPrototypes$(\mathbf{X}, \mathbf{y}, k)$
  $\mathbf{r} \leftarrow$ zeros $(1, n)$ //Create vector to store region identifiers
  $\mathbf{u} = \emptyset$
  for all $i \in \{1 \dots n\}$ **do**
    $a \leftarrow \underset{\forall_{\mathbf{p}_j in \mathbf{P}_P}}{argmin} \left(D\left(\mathbf{p}_j, \mathbf{x}_i\right)\right)$ //Get nearest positive samples
    $b \leftarrow \underset{\forall_{\mathbf{p}_j in \mathbf{P}_N}}{argmin} \left(D\left(\mathbf{p}_j, \mathbf{x}_i\right)\right)$ //Get nearest negative samples
    $z \leftarrow$ CantorPairingFunction(a, b)
    $\mathbf{u} = \mathbf{u} \cup z$ //Set of unique regions
    $r_i \leftarrow z$ //Assign i'th vector to region z
  end for
  $s \leftarrow$ GetRegionStatistics$(\mathbf{X}, \mathbf{y}, \mathbf{r})$
  $z \leftarrow$ GetIncorrectRegion(s)
  while $z \neq \emptyset$ **do**
    $\mathbf{g}\left[\mathbf{g} = z\right] \leftarrow \emptyset$ //Set all values in **g** equals z to $None$
    $\mathbf{u} \leftarrow \mathbf{u} \setminus z$
    $\mathbf{X}' \leftarrow$ UnassignedSamples$(\mathbf{X}, \mathbf{r})$
    $\mathbf{r}' \leftarrow$ AssignRegion$(\mathbf{X}', \mathbf{u}, \mathbf{P}_P, \mathbf{P}_N)$
    $\mathbf{r} \leftarrow$ Update$(\mathbf{r}, \mathbf{r}')$
    $s \leftarrow$ GetRegionStatistics$(\mathbf{X}, \mathbf{y}, \mathbf{r})$
    $z \leftarrow$ GetIncorrectRegion(s) //Get region which doesn't fulfil given statistics or $\emptyset$
  end while
  return r,**u** //Return array assigning each training vector to a particular region
  //and a set of regions.
end function

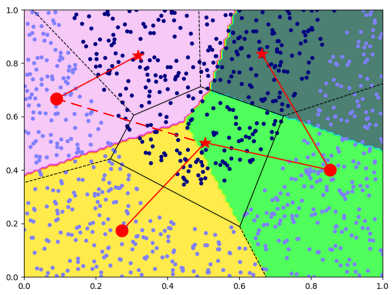


Fig. 1: A visualization of the PPD algorithm showing training samples (purple and dark blue dots), prototypes (red markers), pairs (linked prototypes), and the obtained regions encoded by the background color.

algorithm. Next the regions with the fewest samples and highest unbalanced ratio are pruned. The procedure is repeated until all regions satisfy the given properties.

Next, the process of training the decision trees for each region begins. Each of the decision trees is trained on the subset (region) of the original training set. The size of the regions scales inversely with the number of regions K. The total computational complexity of the proposed method is the sum of the complexities of the PPD algorithm ($O(n)$) and of the decision trees trained on subsets of the samples. On average it is $O\left(n + n \log\left(\frac{n}{K}\right)\right)$.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_11

Algorithm 2 An AssignRegion algorithm used for prediction and reassigning regions which do not fulfill statistical requirements.

```
function AssignRegion(X, u, P_P, P_N)
    n ← ||X||
    for all i ∈ {1 ... n} do
        for all u ∈ u do
            (p_P, p_N) ← CantorUnPairingFunction(u) //Get prototypes defining region u
            d ← D(p_P, x_i)^2 + D(p_N, x_i)^2 //Get distance to pair of prototypes
            if d < d_min then //Find smallest distance
                d_min ← d
                z ← u
            end if
        end for
        g_i ← z //Assign i to region z
    end for
    return g
end function
```

Visualization of the PPD algorithm is presented in figure 1. After defining the regions that are stored in $\mathbf{u}$ (here we have $K = \|\mathbf{u}\|$ regions), a small decision tree is trained using samples within the region. This can be parallelized and executed for each region independently. As the regions contain a relatively small subset of samples, a fragment of the decision boundary which is contained inside a region can be approximated with a simple set of rules. This ensures that the rules are clear and easy to interpret, usually consisting of just a few premises. Since a decision tree is one of the most popular methods for rule extraction, supporting very good scalability, it becomes the first choice for our experiments, but other rule extraction methods can also be used.

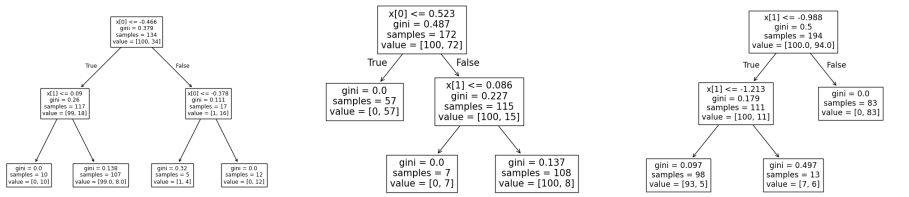
During prediction, PPD starts by identifying a region for each test sample using the *AssignRegion* function presented in sketch 2. The arguments of this function are: a set of test samples $\mathbf{X}$, a set of region identifiers $\mathbf{u}$, and a position of prototypes $\mathbf{P}_P, \mathbf{P}_N$. Then, the main procedure starts by iterating over all test samples. Next, it iterates over a set of possible region identifiers $\mathbf{u}$. Each region identifier is then decoded to the corresponding prototypes $\mathbf{P}_P, \mathbf{P}_N$ using the inverse Cantor function. Then, the nearest pair is determined by the minimum sum of squared distances to the positive and negative prototypes of the pair. Finally, when all samples in $\mathbf{X}$ have a pair of prototypes assigned, the final classifier starts. In a loop, all samples that share the same region are selected and the decision tree related to that particular region is applied to that samples.

3 PPD with the decision trees. Toy example

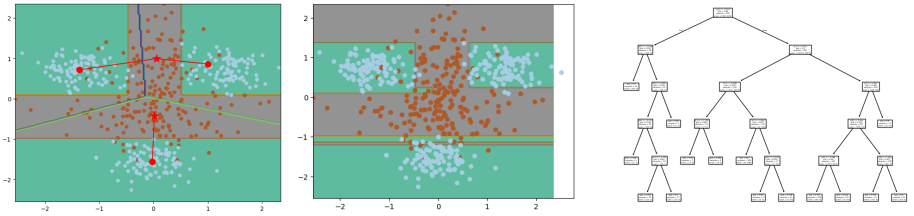
To better explain the PPD algorithm, a toy example is presented, where the PPD algorithm is applied to a binary classification problem.

Figure 2d shows the obtained decision boundary where the input space is divided into 3 regions marked with straight lines. Within each region, a simple decision tree is constructed. In this example, we set the maximum depth of the tree to 2. As a result, three trees are created, as shown in Figure 2a,2b,2c. To show the benefits of the proposed algorithm over the classical decision tree, the decision tree that achieves a comparable prediction performance is shown

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_11



(a) Region 1 (b) Region 2 (c) Region 3



(d) PPD with DT (e) DT without PPD (f) The decision tree

Fig. 2: A comparison of the decision borders and the trees obtained with PPD (a),(b),(c),(d) algorithm and without PPD algorithm (e),(f)

in Figure 2e and its decision border is visualized in Figure 2f. Comparison of these two examples clearly shows that the rules generated by PPD regions with decision trees are much simpler and easier to understand.

4 Experiments

To verify the performance of the PPD algorithm with internal decision trees, we evaluated its prediction accuracy as a function of the depth of internal PPD trees vs. the accuracy of classical decision trees and the influence of PPD parameters on its prediction accuracy using a 10-fold cross-validation.

The software used for the experiments was created in Python, Scikit-learn [9], imbalanced-learn libraries [5] and our own PPD library. It is available at `github.com/mblachnik/Prototype-Pair-Ensemble/tree/24_PPD_Tree`. The datasets obtained from `www.openml.org` present complex two-class classification problems with a non-linear decision boundaries.

4.1 PPD vs Standard Decision Tree. Tree Depth Comparison

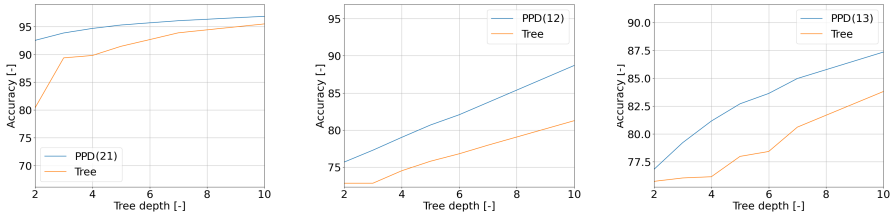
In this section we evaluate the dependencies between size and accuracy of standard decision trees vs. of PPD with small trees inside the PPD regions.

Table 1 shows that for the same accuracy of both methods, standard decision trees on average required 8.56-9.88 premises of a rule, while PPD with trees required only 3-6 rule premises. Moreover, the number of rules in standard trees varies between 185 and 882, while for PPD with trees only 7 to 26 rules are required, with additional prototype-based rule for region identification.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_11

Table 1: Datasets used in the experiments

				Decision tree				PPD(Tree)				
dataset	samples	attr.	unb. ratio	Acc	Std	depth	leaves	Acc	Std	depth	leaves	# reg.
codrnaNorm	488565	8	0.5	95.48	0.08	9.88	882	95.05	0.11	4.00	15.18	33
electricity-normalized	45312	8	0.738	83.77	0.30	9.50	550	83.67	0.59	5.00	26.20	25
covtype	581012	54	0.952	81.25	0.13	9.73	691	81.23	0.71	5.00	26.83	12
banana	5300	2	0.813	88.77	1.20	8.92	237	88.75	1.27	3.00	7.33	6
ring	7400	20	0.981	85.07	0.79	8.56	185	84.31	1.77	6.00	23.71	14
twonorm	7400	20	0.998	84.61	1.45	8.86	344	83.31	1.98	5.00	25.70	10



(a) CodrnaNorm (b) Covtype (c) Electricity-normalized

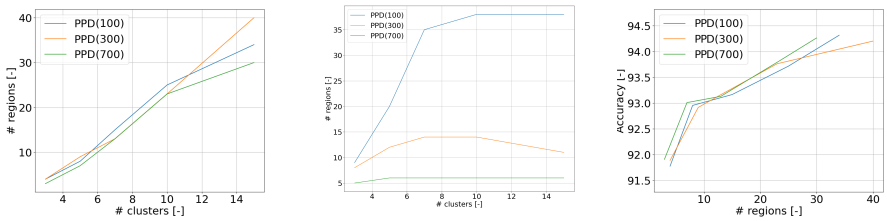
Fig. 3: Relation between the prediction performance and depth of the tree. The number of regions for PPD is given in brackets

To obtain accuracy for various tree depths PPD was configured as follows: minimum samples per region: 400, unbalanced rate: 0.2, and prototypes obtained with k-means clustering with 10 clusters per class (in total 20 prototypes were selected). The obtained results are shown in Figure 3. It can be seen that PPD allows for a significantly simpler representation of the rule base. As trees are constructed inside the PPD regions, they can be much simpler. The maximum number of leaves in a decision tree is $2^{depth+1}$. As can be observed, PPD with local trees of depths 2 to 3 and 10 regions allows for an equivalent accuracy to standard trees of depth 6 to 7.

4.2 Influence of PPD Parameters on its Performance

First, the influence of the number of clusters on the number of regions was analyzed. The obtained results can be divided into two sets of similar behavior: large datasets and small datasets. For large datasets (CodrnaNorm, Electricity-normalized, Covtype) the number of clusters scales linearly with the number of regions, almost independently of the value of $min\_support$. An example for the CordaNorm dataset is shown in Figure 4a. For smaller datasets low $min\_support$ leads to a higher number of regions, while large $min\_support$ merges the smallest regions into the larger ones, and the final number of obtained regions gets fixed as shown in Figure 4b for Ring dataset.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_11



(a) Clusters-Regions large (b) Clusters-Regions small (c) Regions-Accuracy

Fig. 4: The influence of model parameters on its performance. 4a Relation between the number of clusters and the number of obtained regions for CordaNorm dataset - an example of a large dataset. 4b the influence of the number of clusters on the number of regions for the Ring dataset - an example of a small dataset. 4c relation between the prediction performance and the number of regions for the CordaNorm (large) dataset. In the legend, in brackets is given the value of *min_support*.

Second, the influence of the number of regions on model prediction performance was analyzed. Similarly, here, different behavior was observed with respect to the size of the dataset.

For larger datasets, we observed an increase in prediction performance with the growth of the number of regions, as this allows better prototypes to be found and small regions to be removed, as shown in Figure 4c. Consequently, a higher accuracy can be achieved. For smaller datasets, the relation between the number of regions and the prediction performance is less regular and leads to classification performance fluctuations because the number of regions is almost constant.

5 Conclusions

The proposed prototype-pairs decomposition (PPD) algorithm combines case-based reasoning with local rule extraction via in-region decision trees. It produces simpler rule sets than standard decision trees and balances global and local interpretability. This hybrid approach offers an advantage over purely global methods, especially on large datasets with complex decision boundaries.

The application of PPD to multiple class problems, to high dimensional spaces and to models other than decision trees will be the subject of our future research.

Acknowledgments. The research was supported by the Excellence Initiative – Research University program implemented at the Silesian University of Technology, year 2024, project number 11/040/SDW/10-21-01 and the research project BK-227/RM4/2025 funded by the Silesian University of Technology

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_11

References

1. Blachnik, M., Duch, W.: Lvq algorithm with instance weighting for generation of prototype-based rules. Neural Networks **24**(8), 824–830 (2011)
2. Blachnik, M., Kordos, M., Duch, W.: Extraction of prototype-based threshold rules using neural training procedure. In: ICANN 2012, Lausanne, Switzerland, 2012. pp. 255–262. Springer (2012)
3. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE transactions on pattern analysis and machine intelligence **34**(3), 417–435 (2012)
4. Kolodner, J.: Case-based reasoning. Morgan Kaufmann (2014)
5. Lemaître, G., Nogueira, F., Aridas, C.K.: Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. Journal of Machine Learning Research **18**, 1–5 (2017), http://jmlr.org/papers/v18/16-365.html
6. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Advances in neural information processing systems **30** (2017)
7. M. Chakraborty, S. K. Biswas, B.P.: Rule extraction from neural network trained using deep belief network and back propagation. Knowledge and Information Systems **62**, 3753–3781 (2020). https://doi.org/doi:10.1007/s10115-020-01473-0
8. Ma, C., Zhao, B., Chen, C., Rudin, C.: This looks like those: Illuminating prototypical concepts using multiple visualizations. Advances in Neural Information Processing Systems **36** (2024)
9. Pedregosa, F.e.a.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
10. Ribeiro, Marco Tulio, S.S.: Why should i trust you?: Explaining the predictions of any classifier. In: 22nd ACM SIGKDD (2016)
11. Sara El Mekkaoui, Loubna Benabbou, A.B.: Rule-extraction methods from feedforward neural networks: A systematic literature review. https://arxiv.org/html/2312.12878v1 (2023)
12. Saranya A., S.R.: A systematic review of explainable artificial intelligence models and applications: Recent developments and future trends. Decision Analytics Journal **7**, 100230 (2023)
13. Shankaranarayana, S.M., Runje, D.: Alime: Autoencoder based approach for local interpretability. In: IDEAL. pp. 454–463. Springer (2019)
14. Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. In: International conference on machine learning. pp. 3145–3153. PMLR (2017)

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_11