Interval pattern structures for interpreting K-nearest neighbor approach in lazy classification

Position Paper

Alan Tomat[1]

[1]Moscow Institute of Physics and Technology, Institutskiy per. 9, Dolgoprudny, Moscow Region 141700, Russia

Abstract

This paper proposes IPS-KNN, an interpretable variant of the distance-weighted K-nearest neighbors (KNN) algorithm based on interval pattern structures, to address the limitation of KNN's lack of interpretability. The proposed algorithm provides a reason for the classification in the form of a pattern of the interval pattern structure describing the dataset. The intervals in the reason for classification provide insights into which features are important for the classification task and what values these features should have to produce a specific classification result. The IPS-KNN algorithm was evaluated on the red wine quality dataset, where it performed similarly to the distance-weighted KNN algorithm in terms of classification performance. The proposed algorithm can be used in applications where interpretability is important, such as in medical diagnosis or credit risk assessment.

Keywords

lazy classification, pattern structures, interval pattern structures, interpretable K-nearest neighbors

1. Introduction

Lazy learning, also known as instance-based learning, is a type of machine learning algorithms that does not explicitly train the model. Instead, it stores all the training data and uses a similarity measure between the training data and the new data to make predictions [1]. The term "Lazy learning" was first introduced in 1991 in [2]. It was defined as a type of machine learning algorithms that postpones the training process to the testing phase. That is until a new instance is provided for classification. The majority of lazy classification algorithms are K-Nearest Neighbors (KNN) algorithms [3]. The results of KNN algorithms are not inherently interpretable. When both K and the number of features in the dataset are small, visualizing the K-nearest neighbors can provide some sense of interpretability, but not a formal one.

The interpretability of machine learning models lacks a precise mathematical definition. However, there are several non-mathematical definitions that have been proposed. One such definition, stated by Miller in [4], is that interpretability refers to "the degree to which a human can understand the cause of a decision". The researchers in [5] defined interpretable models as

Published in Sergei O. Kuznetsov, Amedeo Napoli, Sebastian Rudolph (Eds.): The 11th International Workshop "What can FCA do for Artificial Intelligence?", FCA4AI 2023, co-located with IJCAI 2023, August 20 2023, Macao, S.A.R. China, Proceedings, pp. 17–24.

alantomat1998@gmail.com (A. Tomat)

0009-0005-6641-2870 (A. Tomat)

those for which humans can understand the causes of the model's predictions.

While interpretability may not be crucial in low-risk applications such as recommendation systems, it is of utmost importance in high-risk applications such as credit risk assessment and medical diagnosis. In such cases, it is essential to pair each prediction with the reason that led the model to make that prediction in order to avoid any possible errors. Despite the absence of a standardized metric for evaluating interpretability, models can be compared based on how easily humans can comprehend the reasons behind their decisions [6].

Pattern structures, an extension of Formal Concept Analysis (FCA) introduced by Kuznetsov and Ganter [7], expands on FCA by enabling the analysis of complex descriptors. Pattern structures, among other generalizations of FCA, have been widely adopted in diverse applications such as information retrieval, web and ontology engineering, biclustering and recommendation, databases and functional dependencies, and software engineering [8].

Interval pattern structures, proposed in [9], are a special case of pattern structures in which the descriptors are numeric intervals. They enable the use of FCA-based knowledge discovery tools on numeric datasets [10]. To classify new instances, pattern structures can be used to extract hypotheses from a set of training instances [11]. The original hypotheses-based lazy classification algorithm on pattern structures was proposed in [12]. This algorithm finds the similarity between a query instance and each instance in the training set, and considers a similarity to be a hypothesis only if it describes instances from a single class. The extracted hypotheses describe the characteristics that the new instance shares with a subset of instances from the training set, providing interpretability into the model's predictions.

Although the original algorithm was proven effective on pattern structures with graph descriptors [11], it suffered from too specific hypotheses in the case of numeric features, especially when the number of features increased [13]. Too specific hypotheses describe only few instances and do not convey much useful information for the classification process. To address this issue, [13] proposed randomly sampling batches of instances from each class in the training set and finding the similarity between the query instance and the entire batch.

Instead of attempting to solve the issues with the original hypotheses-based algorithm, this paper argues that augmenting the distance-weighted KNN with interval pattern structures can provide interpretability. As the nearest instances to the query instance carry the most useful information for classification, this approach focuses on leveraging this information, rather than relying on random batches from the train set.

2. Formal Definitions of Pattern Structures and Interval Pattern Structures

2.1. Pattern Structures

Pattern structures are a generalization of Formal Concept Analysis (FCA). In FCA, a context is defined as a set of instances with binary features, where a binary relation specifies which features are possessed by each instance [14]. Pattern structures remove the binary features condition and allow the use of any type of features as long as they form a lower semi-lattice. We follow [7] to provide a formal definition: Let G be a set of instances, $(D, \sqcap)$ a lower semi-lattice

of all possible instance descriptors, and $\delta : G \to D$ a mapping that corresponds each instance $g \in G$ with its descriptor $d \in D$.

Then $(G, \underline{D}, \delta)$ is called a *pattern structure*, where $\underline{D} = (D, \sqcap)$, with the condition that the set $\delta(G) := \{\delta(g) | g \in G\}$ generates a complete sub-semi-lattice $(D_\delta, \sqcap)$ from $(D, \sqcap)$, i.e. each subset E of $\delta(G)$ has a meet $\sqcap E$ in $(D, \sqcap)$.

The elements of D are called *patterns* and are naturally ordered by the *absorption* relation $\sqsubseteq$: for $c, d \in D$, $c \sqsubseteq d \iff c \sqcap d = c$. The operation $\sqcap$ is also called the *similarity operation*.

The pattern structure $(G, \underline{D}, \delta)$ follows FCA and defines two *derivative operators* $(.)^\diamond$ given in equations 1 and 2. The first of which is applied to a set of instances and returns the largest common pattern describing these instances, while the second is applied to a pattern and returns a set of instances that possess this pattern.

$$A^\diamond = \sqcap_{g \in A} \delta(g) \quad \text{for} \quad A \subseteq G \tag{1}$$

$$d^\diamond = \{g \in G | d \sqsubseteq \delta(g)\} \quad \text{for} \quad d \in (D, \sqcap) \tag{2}$$

2.2. Interval Pattern Structures

Interval pattern structures [9, 10] are a type of pattern structures in which the descriptors of the set D are represented as p-dimensional vectors of intervals, with each interval corresponding to the value of one feature. Here, p is the number of features being analyzed. Interval pattern structures can be used to represent numeric features, by considering each numeric value x as a zero-length interval $[x, x]$.

Thus, the patterns of the set D are $p-$dimensional vectors of intervals, where p is the number of features. For the patterns $e = \langle [a_i, b_i] \rangle_{i=1,2,..,p}$ and $f = \langle [c_i, d_i] \rangle_{i=1,2,..,p}$ in D, the similarity of e and f is given by:

$$e \sqcap f = \langle [a_i, b_i] \rangle_{i=1,2,..,p} \sqcap \langle [c_i, d_i] \rangle_{i=1,2,..,p} = \langle [min(a_i, c_i), max(b_i, d_i)] \rangle_{i=1,2,..,p} \tag{3}$$

The absorption relation $\sqsubseteq$ is given by:

$$\begin{aligned} e \sqsubseteq f &\iff \langle [a_i, b_i] \rangle_{i=1,2,..,p} \sqsubseteq \langle [c_i, d_i] \rangle_{i=1,2,..,p} \\ &\iff [a_i, b_i] \sqcap [c_i, d_i] = [a_i, b_i] \ \ \forall i \in \{1, 2, .., p\} \end{aligned} \tag{4}$$

Consequently, larger intervals are absorbed by the smaller intervals they contain. At first glance, this may seem counter-intuitive, but by definition, the similarity of a set of instances should be absorbed by the pattern of each instance in this set.

3. KNN-based Interval Pattern Structure Lazy Classifier

The original distance-weighted K-nearest neighbors (KNN) algorithm [1] classifies a query instance by finding its K-nearest neighbors and then computing a score for each class based on the inverse of the distance between the query instance and the instances from the K-nearest neighbors that belong to that class. However, the results of KNN algorithms are not inherently

interpretable. To address this limitation, we propose IPS-KNN (Interval Pattern Structure K-Nearest Neighbor): an interpretable variant of the distance-weighted KNN algorithm that is based on interval pattern structures.

3.1. IPS-KNN Lazy Classifier

To classify a new instance g_τ, the set $G_{knn} \subseteq G$ consisting of the k-nearest neighbors of g_τ in the training set G is identified. The similarity of all instances in G_{knn} is then calculated by applying the similarity operator $\sqcap$. All instances in the training set that are described by this similarity are included in a voting process. The class of g_τ is then determined by the function:

$$y(g_\tau) = \operatorname{sgn}(s_+ - s_-) \tag{5}$$

where the positive and negative scores, s_+ and s_- respectively, are calculated as:

$$s_+ = \sum_{g_+ \in G_{knn}^{\diamond\diamond} \cap G_+} \frac{1}{d(g_+, g_\tau)} \tag{6}$$

$$s_- = \sum_{g_- \in G_{knn}^{\diamond\diamond} \cap G_-} \frac{1}{d(g_-, g_\tau)} \tag{7}$$

Here, $d(a, b)$ is the Euclidean distance between instances a and b in the feature space. The set $G_{knn}^{\diamond\diamond}$ is the subset of G that contains all instances described by the similarity $G_{knn}^{\diamond}$.

The proposed IPS-KNN algorithm differs from the original distance-weighted k-Nearest Neighbors in that, in IPS-KNN, the similarity of the k-Nearest Neighbors defines which instances will vote, while in KNN, only the k-nearest neighbors vote. The set of voters in IPS-KNN contains all the k-nearest neighbors because they are all described by their similarity, but it may include additional instances as well.

The difference in the set of voters between IPS-KNN and KNN is due to the geometric shape that encloses the neighbors in the feature space. In KNN, the distance between the query instance and the farthest instance of the k-Nearest Neighbors defines a hyper-sphere in the feature space, and all the instances located on its surface or inside it vote. On the other hand, the similarity of the k-Nearest Neighbors of the query instance defines a hyper-rectangle in the feature space in IPS-KNN. This hyper-rectangle is larger than the hyper-sphere of KNN and contains it entirely within its boundaries. Therefore, the set of voters in IPS-KNN may include more instances than just the k-Nearest Neighbors.

3.2. IPS-KNN Interpretability

The similarity operation $\sqcap$ can reveal why a query instance was classified in a certain class, as it captures the similarity of descriptions. Suppose that a query instance g_τ was classified as positive by IPS-KNN. In this case, $G_{knn}(g_\tau)$ is the set of k-nearest instances to g_τ in the training set, and their common similarity determines the set of voters $V(g_\tau) = G_{knn}^{\diamond\diamond}(g_\tau)$, which can be split into positive voters V_+ and negative voters V_-. The negative score is then given by $s_- = \sum_{g_- \in V_-} \frac{1}{d(g_-, g_\tau)}$. Since g_τ was classified as positive, the positive score s_+ is greater

than s_-. It is possible that only a subset of V_+ needs to vote to classify g_τ as positive; any subset of V_+ that produces a positive score greater than the negative score will lead to the same classification result.

Formally, let $F = \{E \mid E \subseteq V_+,\ \sum_{g_+ \in V_+} \frac{1}{d(g_+, g_\tau)} > s_-\}$ be the set of all subsets of $V_+(g_\tau)$ that give a positive score greater than s_-. The subset with the minimum number of instances includes the instances of V_+ closest to the query instance g_τ. However, this subset is not necessarily unique, since multiple instances might have the same distance from the query instance. Let E be the subset that correspond to the hyper-rectangle with minimum volume. The similarity of the elements of E together with the description of the query instance is the reason for classifying g_τ as positive. This is given by $R = E^\diamond \sqcap \delta(g_\tau) \ : \ E \in F$ and $|E| = \min |B| : B \in F$ and $\Pi_{i=1}^{p} |b_i - a_i|_i$ is minimum, where a_i and b_i are the limits of the i_{th} interval of $E^\diamond \sqcap \delta(g_\tau)$ and p is the number of features. The reason for classification R is itself a pattern in the interval pattern structure $(G, \underline{D}, \delta)$, consisting of a vector of intervals with one interval for each feature. The values of these intervals, relative to the original range of values of the features, provide interpretability. R can also be considered a classifier that correctly classifies all instances of E and the new instance g_τ. It should be noted that substituting the subset E with any other element in F would provide another correct interpretable classifier. However, using E corresponds to enclosing the classified instances in the most specific hyper-rectangle. On the other hand, using V_+ instead of E will result in the most general hyper-rectangle.

The interpretability provided by IPS-KNN is somewhat analogous to that provided by decision trees for continuous features. In IPS-KNN, the reason for classification is represented by a hyper-rectangle in the feature space, with one interval for each feature. Similarly, in decision trees, decisions are based on comparisons of feature values using greater than and less than conditions [15], which can also be represented as hyper-rectangles, with two exceptions. Firstly, decision trees may not use all features in the classification process, resulting in intervals of the form $(-\infty, +\infty)$. Secondly, decision trees may bound the values of some features from one side only, leading to intervals of the form $(-\infty, x)$ and $(x, +\infty)$, where x is a real value. Section 4 further explains how these intervals are used for interpretability.

4. Experiments

The proposed algorithm IPS-KNN was evaluated against the baseline classification algorithms on the red wine quality dataset [1], which contains 11 numeric features and its output feature was binarized for this purpose. In its original form, the quality of each wine is assessed by a score between 1 and 10, where a higher score indicates better quality. To binarize the output feature, wines with a quality score in the range $[1, 5]$ were considered 'bad', while those in the range $[6, 10]$ were considered 'good'. To assess the performance of the models, 20% of the instances in the dataset were randomly selected as a holdout test set, and the remaining 80% were used as the training set. In addition, 5-fold stratified cross-validation was used to evaluate the models and to tune their hyperparameters using a grid search approach. F1 score was used as the evaluation metric to take class imbalance into account. The accuracy and F1 scores on the held-out test set for the proposed algorithm, distance-weighted KNN, and other baseline

[1] https://archive.ics.uci.edu/ml/datasets/wine+quality

Table 1
Accuracy and F1 for IPS-KNN and the baseline classification algorithms on the binarized red wine quality dataset.

Classifier	Accuracy	F1
KNN	82.81%	83.97%
IPS-KNN	81.56%	82.8%
Naive Bayes (GaussianNB)	75%	76.05%
Logistic Regression	77.5%	78.44%
SVM	78.75%	80.68%
Decision Tree	77.5%	78.95%
Random Forest	84.06%	84.59%
XGBoost	84.69%	86.04%

classification algorithms are presented in Table 1. The evaluation results indicate that the IPS-KNN algorithm and the distance-weighted KNN algorithm had similar performance. This suggests that our proposed modification to the KNN algorithm, which allows more instances to vote, did not significantly degrade performance. This is because the K nearest neighbors, which have the largest voting weight, are still included in the set of voters. Table 1 also shows that Random Forest and XGBoost outperformed IPS-KNN, with XGBoost achieving the highest F1 score with a 3.26% increase over that of IPS-KNN. Both XGBoost and Random Forest use decision trees, which means that their results can be interpreted using the tests performed at the nodes along the paths through the structure used to classify a query instance. However, after grid search, the number of decision trees used by Random Forest and XGBoost was 100 and 1000, respectively. This large number of decision trees makes direct interpretability a difficult task.

The following is an example of how IPS-KNN classifies a new instance and provides interpretability on the binary red wine quality dataset. Table 2 shows the features of the red wine quality dataset, the values of these features for the query instance g_τ, the ranges of the features, and the intervals of the reason of classification R. We obtained the value of the hyperparameter K of 25 through grid search. The similarity of 25-nearest instances to g_τ chose 44 instances to vote, 38 of which are positive and the remaining 6 are negative. The resulting positive and negative scores were $s_+ = 19.3$ and $s_- = 2.6$; therefore, g_τ was classified as positive.

Out of the 38 positive instances, the nearest 4 were enough to give a positive score larger than the negative one. The similarity of these 4 objects together with g_τ produced the reason of classification denoted by R in Table 2. The intervals of R provide the interpretability of why g_τ was classified as positive and what values should the features of instances similar to g_τ also have in order to be classified as positive.

For example, the interval $[6.3, 6.8]$ of R corresponding to the feature (fixed acidity) shows that this feature's value should be low relative to the original range of values in order to the classification result to be positive. The interval corresponding to the feature (alcohol) shows that the value of this feature should be in in the middle of the range of possible values. The intervals corresponding to the features (fixed acidity, residual sugar, sulphates) are short relative to the ranges of their values, which mean that they are important to the classification process.

Table 2
Ranges of data set feature values (red wine quality) and interval classifier intervals.

Feature	Feature values of g_τ	Range of values	Reason of classification R
Fixed acidity	6.3	$[4.6, 15.9]$	$[6.3, 6.8]$
Volatile acidity	0.55	$[0.1, 1.6]$	$[0.49, 0.67]$
Citric acid	0.15	$[0, 1]$	$[0.02, 0.22]$
Residual sugar	1.8	$[0.9, 15.5]$	$[1.8, 2.3]$
Chlorides	0.077	$[0.01, 0.61]$	$[0.061, 0.077]$
Free sulfur dioxide	26	$[1, 72]$	$[13.0, 37.0]$
Total sulfur dioxide	35	$[6, 289]$	$[24.0, 53.0]$
Density	0.99314	$[0.99, 1.004]$	$[0.99314, 0.99489]$
pH	3.32	$[2.7, 4]$	$[3.32, 3.41]$
Sulfates	0.82	$[0.3, 2]$	$[0.76, 0.83]$
Alcohol	11.63	$[8.4, 14.9]$	$[10.3, 11.6]$

In contrast, intervals covering a large part of the entire range of values are less useful since they describe almost all objects in the dataset.

5. Conclusion

Based on the results of the experiments, we can conclude that the IPS-KNN algorithm performs similarly to the distance-weighted KNN algorithm in terms of classification performance. However, the IPS-KNN algorithm provides additional interpretability through the reason for the classification in the form of a pattern of an interval pattern structure. The intervals in the reason for classification provide insights into which features are important for the classification task and what values these features should have in order to produce a specific classification result.

The proposed algorithm can be used in applications where interpretability is important, such as in medical diagnosis or credit risk assessment. The interpretability provided by IPS-KNN can help experts understand why a certain decision was made, which can lead to better decision making and increased trust in the system.

6. Future Work

One limitation of the IPS-KNN algorithm is that the hyper-rectangle surrounding the query instance and its neighbors is currently aligned with the coordinate axes in the feature space. To address this limitation, we plan to investigate the introduction of a tilted hyper-rectangle in future work. By tilting the hyper-rectangle at an angle, we aim to provide better separation of the voters and potentially improve classification accuracy.

Introducing a tilted hyper-rectangle is a promising avenue for future research that could enhance the performance of the IPS-KNN algorithm. However, it is important to conduct more detailed investigation to explore the feasibility and potential benefits of this approach. Specifically, we need to determine the optimal angle of tilt and evaluate the impact of this approach on classification accuracy compared to the current IPS-KNN algorithm.

References

[1] D. W. Aha, D. Kibler, M. K. Albert, Instance-based learning algorithms, Machine learning 6 (1991) 37–66.

[2] D. G. Lowe, Learning in Autonomous Agents: Exploration, Curiosity, and Learning in the Absence of External Rewards, Ph.D. thesis, Department of Computer Science, University of British Columbia, 1991.

[3] D. Wettschereck, D. W. Aha, T. Mohri, A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms, Artificial Intelligence Review 11 (1997) 273–314.

[4] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial intelligence 267 (2019) 1–38.

[5] E. Dudyrev, I. Semenkov, S. O. Kuznetsov, G. Gusev, A. Sharp, O. S. Pianykh, Human knowledge models: Learning applied knowledge from the data, Plos one 17 (2022) e0275814.

[6] C. Molnar, Interpretable machine learning, Lulu. com, 2020.

[7] B. Ganter, S. O. Kuznetsov, Pattern structures and their projections, in: Conceptual Structures: Broadening the Base: 9th International Conference on Conceptual Structures, ICCS 2001 Stanford, CA, USA, July 30–August 3, 2001 Proceedings 9, Springer, 2001, pp. 129–142.

[8] S. Ferré, M. Huchard, M. Kaytoue, S. O. Kuznetsov, A. Napoli, Formal concept analysis: from knowledge discovery to knowledge processing, A Guided Tour of Artificial Intelligence Research: Volume II: AI Algorithms (2020) 411–445.

[9] S. O. Kuznetsov, Fitting pattern structures to knowledge discovery in big data, in: Formal Concept Analysis: 11th International Conference, ICFCA 2013, Dresden, Germany, May 21-24, 2013. Proceedings 11, Springer, 2013, pp. 254–266.

[10] M. Kaytoue, S. O. Kuznetsov, A. Napoli, S. Duplessis, Mining gene expression data with pattern structures in formal concept analysis, Information Sciences 181 (2011) 1989–2001.

[11] S. O. Kuznetsov, Scalable knowledge discovery in complex data with pattern structures, in: Pattern Recognition and Machine Intelligence: 5th International Conference, PReMI 2013, Kolkata, India, December 10-14, 2013. Proceedings 5, Springer, 2013, pp. 30–39.

[12] B. Ganter, S. O. Kuznetsov, Hypotheses and version spaces, in: ICCS, volume 2746, Springer, 2003, pp. 83–95.

[13] A. Masyutin, Y. Kashnitsky, S. Kuznetsov, Lazy classification with interval pattern structures: Application to credit scoring, in: Workshop Notes, 2015, p. 43.

[14] B. Ganter, R. Wille, Formal concept analysis: mathematical foundations, Springer Science & Business Media, 2012.

[15] B. De Ville, P. Neville, Decision trees for analytics: using SAS Enterprise miner, SAS Institute Cary, NC, 2013.