Efficient Subgroup Discovery Through Auto-Encoding

Joost F. van der Haar, Sander C. Nagelkerken, Igor G. Smit, Kjell van Straaten, Janneke A. Tack, Rianne M. Schouten, and Wouter Duivesteijn(✉)

Technische Universiteit Eindhoven, Eindhoven, the Netherlands
{j.f.v.d.haar,s.c.nagelkerken,i.g.smit,k.v.straaten, j.a.tack}@student.tue.nl, {r.m.schouten,w.duivesteijn}@tue.nl

Abstract. Current subgroup discovery methods struggle to produce good results for large real-life datasets with high dimensionality. Run times can become high and dependencies between attributes are hard to capture. We propose a method in which auto-encoding is applied for dimensionality reduction before subgroup discovery is performed. In an experimental study, we find that auto-encoding increases both the quality and coverage for our dataset with over 500 attributes. On the dataset with over 250 attributes and the one with the most instances, the coverage improves, while the quality remains similar. For smaller datasets, quality and coverage remain similar or see a minor decrease. Additionally, we greatly improve the run time for each dataset-algorithm combination; for the datasets with over 250 and 500 attributes run times decrease by a factor of on average 150 and 200, respectively. We conclude that dimensionality reduction is a promising method for subgroup discovery in datasets with many attributes and/or a high number of instances.

Keywords: Subgroup discovery · Auto-encoding · Dimensionality reduction

1 Introduction

Subgroup Discovery (SD) is a data mining method used to discover interesting relationships between objects in a dataset with respect to a specific target variable. The SD outcome is typically represented as a set of rules called subgroups [10]. SD methods are often used on real-world problems, such as the detection and description of Coronary Heart Disease risk groups [8], fraud detection in the healthcare domain [16] and identifying flight delay patterns [23]. Real-life problems often involve datasets with high dimensionality. For many SD methods, handling such large datasets can be an issue. The most commonly used method to address this problem is by applying sampling. However, this method

J. F. van der Haar, S. C. Nagelkerken, I. G. Smit, K. van Straaten and J. A. Tack— These authors contributed equally to this work.

T. Bouadi et al. (Eds.): IDA 2022, LNCS 13205, pp. 327–340, 2022.
https://doi.org/10.1007/978-3-031-01333-1_26

has the downside of not taking dependencies and relationships between variables into account which can result in important data loss for subgroup discovery. It seems that no research is conducted on reducing the dimensionality of a dataset via auto-encoding prior to applying subgroup discovery methods. Using auto-encoding as a method to reduce the dimensionality of a dataset may solve the issue of important data loss since this method is able to uncover latent low-dimensional non-linear structures in the data [11], which allows it to minimize the dimensionality reduction information loss. Therefore, this paper investigates the effects of auto-encoding on the results of various existing SD methods.

1.1 Main Contribution

We propose an alternative method that enables the application of SD on larger datasets. We show that preprocessing datasets by performing dimensionality reduction using auto-encoders can improve the efficiency of SD, while maintaining or improving subgroup quality and coverage of discovered subgroups. Run times can be a few hundred times less for datasets with many attributes. At the same time, we can increase the coverage and explore different regions in the data for any algorithm if datasets are reasonably sized. We can do this while achieving equivalent or even higher subgroup qualities, both on average and for the best subgroup, depending on the dataset.

2 Related Work

2.1 Subgroup Discovery

Subgroup Discovery methods (see [10] for a survey) can be partitioned into three groups. The first group of methods are extensions of classification algorithms, such as EXPLORA [14], MIDOS [29], SD [8], and CN2-SD [18]. The second group contains extensions of association algorithms, such as APRIORI-SD [13] and Merge-SD [9]. The third group consists of evolutionary algorithms, such as NMEEF-SD [3]. Herrera et al. [10] noticed that many of the above listed subgroup discovery techniques have difficulties with real-world problems due to high dimensionality of the datasets associated with such problems. Usually, there are two solutions for data mining algorithms that do not perform well under high dimensional datasets, namely reducing the data size without changing the outcome radically or redesigning the algorithm so that it can handle huge datasets. The most applied method to reduce the dimensionality of a dataset is sampling, in which particular instances of a dataset are selected according to certain criteria [10]. A downside of this technique is that this could lead to loss of important knowledge for the SD task when not considering dependencies and relationships between variables. Therefore, when reducing the dimensionality of a dataset, it must be ensured that no important data is lost which is necessary for the extraction of important subgroups [10].

2.2 Dimensionality Reduction

The goal of dimensionality reduction is to produce a compact low-dimensional encoding of a given high-dimensional dataset. Principal Component Analysis (PCA) [28] aims to find a linear subspace of a dimension lower than the dimension of the original dataset, such that the data points lie mainly on this linear subspace, and thus maintain most of the variability of the data [25,27]. Linear Discriminant Analysis (LDA) [24] is a classifier that is used to find a linear combination of features, which separates a number of classes of data. The main idea is to ensure that the samples after projection have maximum between-cluster-distance and minimum within-cluster-distance in the new subspace [27]. Isomap [25] performs multidimensional scaling in the geodesic space of the non-linear data manifold, rather than in the input space. Lastly, auto-encoders [27] reduce dimensionality very well while maintaining more information than the four aforementioned dimensionality reduction methods for most datasets. Additionally, auto-encoders are better capable of detecting repetitive structures than the alternative methods.

3 Preliminaries

A dataset D consists of a set of individuals I and attributes A, such that $D = (I, A)$. A subgroup description, also called a complex pattern P, is a set of selectors, also called basic patterns [2]. For a nominal attribute, a selector is a Boolean function that is true if $a_i \in A = v_j$ for the individual, and false otherwise. For numeric attributes, the value of the selector is set to true for an individual if the attribute value for that individual is in the interval $[\min_j : \max_j]$, thus if $a_i \in [\min_j : \max_j]$, and false otherwise. The set of all basic patterns in the dataset is denoted by Σ. The subgroup description P is then defined by a conjunction of basic patterns: $P(i) = \text{sel}_j \wedge ... \wedge \text{sel}_n$, $\text{sel}_m \in \Sigma$, $m = j, ..., n$ for individual $i \in I$. This pattern can then be interpreted as a rule for a subgroup $S_P := \{i \in I | P(i) = true\}$ [2]. A subgroup S_P is thus defined as the set of all individuals $i \in I$ that satisfy the rule based on the conjunction in P, consisting of a set of selectors.

Subgroup Discovery is a technique for descriptive and exploratory data mining. The goal of SD is to identify subsets of a given dataset that display interesting behaviour [2]. The interestingness of behaviour is defined as "distributional unusualness with respect to a certain property of interest" [29]. To what extent behaviour is interesting, is evaluated with respect to certain interestingness criteria, which are formalized by a quality function. Using this quality function, a subgroup discovery algorithm identifies a set of interesting subgroups. In this paper, we employ Weighted Relative Accuracy (WRAcc) [17] as quality function. The WRAcc of a subgroup is defined in the following way [20]:

$$\text{WRAcc}(S_P) = \frac{|S_P|}{|I|} * (p_{S_P}(\text{target} = 1) - p_I(\text{target} = 1))$$

The task of Subgroup Discovery in this paper now becomes equivalent to the formal problem definition in [6, Problem Statement 1], with $\Omega = D$, $\mathcal{D}$ is as described earlier in this section, $\varphi = \text{WRAcc}$, $q = 100$, and $\mathcal{C} = \varnothing$.

3.1 Auto-Encoding

An auto-encoder is a three-layered neural network, consisting of an encoding layer, an encoded layer, and a decoding layer. The encoding layer takes an individual $i \in \mathbb{R}^d$ as input and reduces it to an item $h \in \mathbb{R}^{d'}$, where typically $d' \ll d$. This layer is subsequently decoded to produce a reconstructed version $i' \in \mathbb{R}^d$ of the individual. The objective of the auto-encoder is then to minimize the reconstruction error $J(i, i') = \frac{1}{2}\|i - i'\|_2^2$ [27], such that this reconstructed version is as close to the original data entry as possible. Given a dataset D, such a network can be trained using backpropagation of the so-called mean squared error, which is the average of this loss over the data in D. This training occurs for a certain number of epochs, which are passes through the dataset. To prevent overfitting, the training can stop earlier once the test error has not improved for a certain number of steps. Once the auto-encoder has completed training, its encoded layer can be used as a dimension-reduced version of the input data.

The structure of auto-encoders can vary with regard to the number of hidden layers, size of the hidden layers, and activation function used in its neurons. Deep auto-encoders tend to perform better than shallower ones with only a single hidden layer [11], although this advantage disappears if the number of free parameters becomes too big as a result [11]. The neurons in the layers can have several activation functions. Often the Leaky ReLU [21] activation function is used, due to strong performance and immunity to the dying neuron problem. The Leaky ReLU activation function is given by:

$$f(x) = \begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

Here, α is a typically small coefficient that is chosen by the user.

4 Methodology

We propose a method of combining auto-encoding with SD. Our original dataset D may contain attributes of any type: binary, nominal, and numeric. Auto-encoding expects input data that is real-valued, and we bridge that gap by one-hot encoding all non-numeric attributes of the original dataset. This results in a new dataset D' whose individuals take value in $\mathbb{R}^d$, and whose dimensionality is larger than the number of attributes in the original dataset: $d \geq |A|$. Subsequently, an auto-encoder neural network is trained on this dataset D' using backpropagation on the mean squared error. This auto-encoder has d' encoded features, where d' is chosen in such a way that it provides a balance between a small number of features and a high representativeness of the features. Once

this auto-encoder is developed, every individual $i \in D'$ is transformed to an item $h \in \mathbb{R}^{d'}$. Using this set of transformed items and associated set of attributes, we can perform SD as described in Sect. 3, directly using this new data in existing SD algorithms.

4.1 Experimental Setup

To conduct our analysis on the effect of auto-encoding on the performance of SD, we perform multiple experiments[1]. These experiments are conducted using several SD algorithms with various datasets (see Sect. 4.3). We test the algorithms both with and without auto-encoding and compare the results. The employed SD algorithms are beam search [6], APRIORI-SD [13], Best First Search (BFS) [31], and Depth First Search (DFS) [20]. These are implemented in Python, using the adapted code from [5] for the beam search algorithm and a modified version of the Python package pysubgroup [20] for the other algorithms.

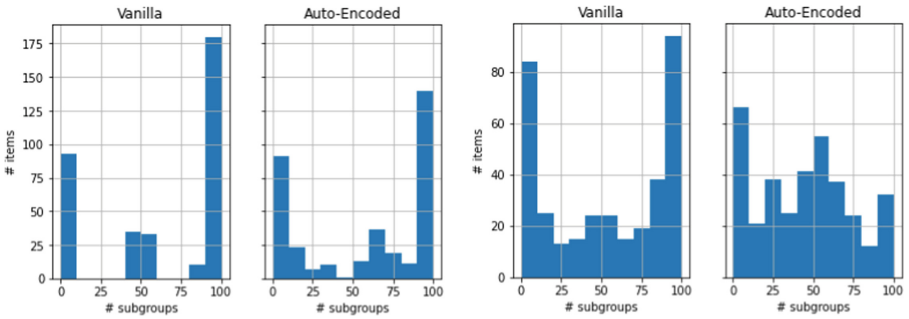
For all datasets, the auto-encoder is implemented using TensorFlow 2.0 [1] in Python. The number of encoded features d' of the auto-encoder is selected individually for each dataset. Here, we choose a number that provides a good balance between the number of features and the error function. The intuition here is similar to that of the elbow rule (or critical point rule) in clustering [26]. The selected values for d' are reported in Table 1.

During the tuning of the number of features, the number of epochs and patience for early stopping are set to 100 and 10, respectively. We set the number of hidden layers before and after the encoded layer to 4 and the number of neurons per layer to 512, 256, 128, and 64 (reversed in the decoder). For the neurons, we use the Leaky ReLU activation function with $\alpha = 0.3$, following the findings of [30].

We evaluate the performance of auto-encoder based SD along three axes. Firstly, to represent subgroup quality, we report the mean and maximum WRAcc for the 100 best-found subgroups. Secondly, to represent dataset coverage, we determine the number of items that are present in at least one subgroup, as well as the number of distinctive items[2] between vanilla and auto-encoder based SD: *added items* are those present in at least one subgroup found through auto-encoder based SD but in none of the subgroups found through vanilla SD, and the reverse are *lost items*. Thirdly, to represent subgroup diversity, we check the distribution of the number of subgroups in which each item is present.

[1] cf. Github repository at https://github.com/JFvdH/Efficient-SD-through-AE.

[2] Notice that, for making these distinctive comparisons, we must compare presence or absence of individuals in subgroups in the original data space, with presence or absence of encoded items in subgroups in the encoded space. Naively, this may seem nontrivial, but notice that the number of individuals and the number of items is identical: when encoding, the *representation* of each individual is changed and its number of attributes may change, but each individual has *one unique counterpart item* in the encoded space. This enables identification of added and lost items across the divide between original data space and encoded space.



(a) Without DSSD. (b) With DSSD.

Fig. 1. Effect of Diverse Subgroup Set Discovery (DSSD) on distributions of item occurrence in subgroups.

4.2 Algorithms and Settings

The SD algorithms that we analyse are beam search [6], APRIORI-SD [13], Best First Search (BFS) [31], and Depth First Search (DFS) [20]. All algorithms are set to find subgroups with a depth of 2, meaning that they have to find subgroups using patterns of at most 2 selectors. All subgroups are parameterized to report the best 100 subgroups found, evaluated by WRAcc. For beam search, additional parameters require configuration: the beam width was set to (a generous) 100, the minimum support for a subgroup to be considered was set to 2% of the dataset, and numeric attributes were treated with the `lbca` discretization method from [22] with granularity 5.

We adapt the SD algorithms to incorporate the lessons learned from Diverse Subgroup Set Discovery (DSSD) [19]. In beam search, a candidate subgroup is discarded unless its quality differs from the quality of its seed subgroup. For the other algorithms, the same principle is implemented in a slightly different way: a candidate is now discarded if its quality is (approximately) equal to any current candidate's quality and all but 1 selectors are identical. Figure 1 illustrates the effect of this DSSD strategy on the distribution of the number of subgroups encompassing items: variety increases under DSSD.

4.3 Data

The six datasets used for this research are extracted from the UCI Machine Learning Repository. Those datasets are selected since they all have different compositions such that a variety of dataset characteristics are tested. An overview of the number of rows, number of attributes, and type of attributes is presented in Table 1.

For the Soybean dataset, all rows with missing data are dropped. This means that N decreases from 307 to 266. In the Arrhythmia dataset, one attribute contained 376 missing values. Instead of removing the majority of our rows, we

Table 1. Metadata (before preprocessing) of datasets used for the experiment.

Dataset	N	#Attributes			d'
		Discrete	Numerical	Total	
Ionosphere	351	0	34	34	5
Soybean(-large)	307	35	0	35	9
Adult	48842	8	6	14	7
Mushroom	8124	22	0	22	5
Arrhythmia	452	73	206	279	8
Indoor	21048	2	520	522	9

chose to drop this attribute, resulting in a remaining 205 numerical attributes in this dataset. After this attribute drop, 32 rows contain further missing values. Those rows are dropped resulting in $N = 420$. The datasets Ionosphere, Adult, and Mushroom did not contain any missing values, so N remains 351, 48842, and 8124, respectively. Lastly, in the Indoor dataset, multiple target variables are present. We select BuildingID as the target of our SD run and drop all other target variables. No rows are dropped, so N remains 21048.

WRAcc evaluation requires one target class per dataset to be designated as the positive class. For Ionosphere we select "good", for Mushroom we select poisonous mushrooms, for Adult we select persons making over 50K a year, for Arrhythmia we select having any heart disease, for Soybean we select soybeans having any "spot" classification, and for Indoor dataset we select all objects having BuildingID '2'.

Lastly, to ensure proper training of the auto-encoder, the attributes in the Adult and Arrhythmia datasets are standardized before auto-encoding. These datasets contain attribute values of significantly varying orders of magnitude. If not standardized, training based on the mean square error becomes both very unstable and skewed towards only those attributes with large values. This leads to poor results of the encoder. Hence, all data entries are standardized by subtracting the sample mean and dividing by the sample standard deviation of the specific attribute.

5 Results

An overview of the results of the algorithms with and without auto-encoding can be seen in Table 2. All auto-encoders had a small number of encoded features compared to the original numbers of features. With this lower dimensionality, depending on the dataset and algorithm, slightly varying results are obtained.

Firstly, in Table 2a we find that the coverage of the discovered subgroups using beam search is increased for every dataset except for the Mushroom dataset (where it stays approximately the same). This observation will be further discussed in Sect. 6. From the inspection of the added and lost items, we see that,

Table 2. Comparative performance of Subgroup Discovery with and without auto-encoding, in terms of runtime, quality, and coverage.

(a) Beam search, using min_sgsize = 2%, n_chunks = 5, beam_width = 100.

Dataset	Vanilla				Auto-encoded						Items	
	Run time (s) (Algorithm)	WRAcc Max	WRAcc Mean	Coverage	Run time (s) Tun.	Run time (s) Enc.	Run time (s) Alg.	WRAcc Max	WRAcc Mean	Coverage	Added	Lost
Ionosphere	396	0.141	0.132	0.858	50	5	41	0.097	0.087	0.883	33 (0.094)	24 (0.068)
Soybean	221	0.250	0.248	0.508	61	4	90	0.166	0.143	0.662	48 (0.180)	7 (0.026)
Adult	945	0.066	0.065	0.458	4810	650	180	0.064	0.062	0.565	7740 (0.158)	2528 (0.052)
Mushroom	309	0.242	0.226	0.566	1184	133	46	0.163	0.153	0.553	993 (0.122)	1096 (0.135)
Arrhythmia	7800	0.085	0.082	0.450	83	6	91	0.084	0.073	0.540	106 (0.252)	68 (0.162)
Indoor	47644	0.117	0.114	0.378	4980	650	187	0.165	0.149	0.463	2972 (0.141)	1191 (0.057)

(b) Best-first search.

Dataset	Vanilla				Auto-encoded						Items	
	Run time (s) (Algorithm)	WRAcc Max	WRAcc Mean	Coverage	Run time (s) Tun.	Run time (s) Enc.	Run time (s) Alg.	WRAcc Max	WRAcc Mean	Coverage	Added	Lost
Ionosphere	0.9	0.069	0.043	0.997	50	5	0.1	0.089	0.021	0.972	1 (0.003)	10 (0.028)
Soybean	0.2	0.241	0.162	0.944	61	4	0.1	0.115	0.033	0.910	13 (0.049)	22 (0.083)
Adult	2.7	0.069	0.030	0.853	4810	321	0.5	0.053	0.011	0.936	5463 (0.112)	1432 (0.029)
Mushroom	0.9	0.182	0.093	1.000	1184	133	0.1	0.113	0.021	0.936	0 (0.000)	522 (0.064)
Arrhythmia	46.9	0.066	0.058	0.705	83	6	0.2	0.058	0.017	0.874	91 (0.217)	20 (0.048)
Indoor	108.7	0.107	0.106	0.400	4980	650	0.4	0.109	0.032	0.900	10530 (0.500)	3 (0.000)

(c) Depth-first search.

Dataset	Vanilla				Auto-encoded						Items	
	Run time (s) (Algorithm)	WRAcc Max	WRAcc Mean	Coverage	Run time (s) Tun.	Run time (s) Enc.	Run time (s) Alg.	WRAcc Max	WRAcc Mean	Coverage	Added	Lost
Ionosphere	1.0	0.069	0.043	0.989	50	5	0.1	0.043	0.015	0.949	15 (0.043)	1 (0.002)
Soybean	0.3	0.241	0.162	0.944	61	4	0.1	0.087	0.027	0.944	15 (0.056)	15 (0.056)
Adult	5.05	0.063	0.029	0.853	4810	321	0.5	0.054	0.011	0.936	5463 (0.112)	1432 (0.029)
Mushroom	1.4	0.182	0.093	1.000	1184	133	0.1	0.120	0.022	0.936	0 (0.000)	522 (0.064)
Arrhythmia	46.6	0.086	0.062	0.681	83	6	0.2	0.083	0.022	0.855	97 (0.231)	24 (0.057)
Indoor	117.1	0.109	0.107	0.350	4980	650	0.5	0.134	0.039	0.909	11752 (0.558)	0 (0.000)

(d) APRIORI-SD.

Dataset	Vanilla				Auto-encoded						Items	
	Run time (s) (Algorithm)	WRAcc Max	WRAcc Mean	Coverage	Run time (s) Tun.	Run time (s) Enc.	Run time (s) Alg.	WRAcc Max	WRAcc Mean	Coverage	Added	Lost
Ionosphere	1.1	0.074	0.046	0.997	50	5	0.3	0.086	0.022	0.972	1 (0.003)	10 (0.028)
Soybean	0.2	0.241	0.162	0.944	61	4	0.1	0.112	0.032	0.914	13 (0.049)	21 (0.079)
Adult	2.6	0.063	0.029	0.853	4810	321	0.6	0.055	0.011	0.936	5463 (0.112)	1432 (0.029)
Mushroom	0.4	0.182	0.093	1.000	1184	133	0.1	0.106	0.021	0.936	0 (0.000)	522 (0.064)
Arrhythmia	39.3	0.084	0.062	0.702	83	6	0.8	0.078	0.022	0.855	90 (0.214)	26 (0.061)
Indoor	94.8	0.129	0.122	0.290	4980	650	1.7	0.116	0.039	0.909	13016 (0.618)	0 (0.000)

besides increasing coverage, beam search with the auto-encoded data covers a different region of the data for most datasets. Some items are newly included in its 100 best subgroups and other items are now excluded. For the Adult dataset, however, very few items are lost while many are added. Thus, here, auto-encoding expands the coverage region.

In Tables 2b, 2c, and 2d, we find consistent effects on the coverage for the other three algorithms. Increased coverage is only achieved for the Adult,

Arrhythmia, and Indoor datasets: the larger ones in terms of items and/or attributes. For the smaller datasets, the vanilla algorithms already achieve such high coverage that auto-encoding can only match the coverage. The coverage of the Adult data (relatively many items) is a few percentage points higher for all three algorithms. Auto-encoding based SD manages to cover 15% points more of the Arrhythmia data (relatively many attributes). Lastly, for the Indoor data (relatively many attributes and items), the coverage is increased by at least 50% points for BFS, DFS, and APRIORI-SD. For these algorithms, the number of items added and lost for the bigger datasets show similarities with the numbers for beam search. Namely, in the Adult and Arrhythmia data, a part of the items is newly included after auto-encoding while another part is excluded. The findings, therefore, cover different regions of the data. For the Indoor data, on the other hand, many items are added to at least one subgroup while few to none are lost for each algorithm. Hence, we find that auto-encoding increases the coverage of all subgroup discovery algorithms significantly for data with a high number of items and/or attributes. It includes different regions of the data or expands the current regions. For smaller datasets, the coverage is approximately equal.

In terms of WRAcc quality of the found subgroups, again, a difference can be seen between the three smaller and three larger datasets. Table 2a displays that the maximum and mean WRAcc after auto-encoding are worse for Ionosphere, Soybean, and Mushroom when performing beam search. Oppositely, the qualities for the bigger Adult and Arrhythmia datasets remained similar after auto-encoding. The subgroups found on the Indoor dataset are substantially better with auto-encoding than without. It is likely that due to the high number of numerical variables, the attributes that are present do not have enough expressive power to form strong groups with a small number of selectors while the encoded attributes do have this expressive power.

For the other algorithms, we find similar results in Tables 2b, 2c, and 2d. An important distinction, however, is that the mean WRAcc scores over the 100 subgroups after auto-encoding are generally lower for all datasets, indicating that the number of high-quality subgroups is lower. From the maxima, though, we can see that the performance with auto-encoding of the three big datasets is good, again. For all three algorithms, the maximum WRAcc scores for the Adult and Arrhythmia data are equivalent with and without auto-encoding. One exception is APRIORI-SD, in which auto-encoding decreases the maximum WRAcc for the Adult data. The maximum WRAcc score for the Indoor dataset is higher with auto-encoding using BFS and DFS and only slightly lower for APRIORI-SD. For the smaller Soybean and Mushroom datasets, again we find that the maximum WRAcc scores decreases with auto-encoding for APRIORI-SD, DFS, and BFS. On the other hand, the subgroups of the Ionosphere dataset, which has a reasonably high number of numerical features, have a higher maximum WRAcc with auto-encoded features for BFS and APRIORI-SD algorithm.

Finally, run times of the algorithms itself on auto-encoded versions of each dataset are improved across all of Table 2. For the Indoor dataset, the time is reduced by a factor of over 250 for multiple algorithms. In Table 2a, we see that

the beam search run time for the Arrhythmia data is decreased from more than 2 h to 91 s and for the Indoor dataset it is even decreased from over 13 h to just 3 min. Of course, this neglects tuning and training time of the auto-encoders: for some combinations of dataset and algorithm this time is longer than the gained algorithm run time. For the combination of Arrhythmia and Indoor with beam search, we see that the gained time is bigger than the total model development time.

6 Discussion

From the results, we can derive that feature reduction using auto-encoding can help to improve subgroup discovery for datasets with many attributes and/or instances. We found that, depending on the algorithm and dataset, the coverage, quality, and run time can be improved by auto-encoding the data. For smaller datasets with fewer attributes, this improvement is smaller. Here, the coverage is often similar and the run time is shorter, but the quality of the subgroups is generally lower. The method could still be used to explore different regions within the data but, in general, the added value is low.

For the datasets that do benefit from auto-encoding, we saw that for some dataset-algorithm combinations, the model development time is higher than the algorithm run time. However, for the largest two datasets in terms of attributes, we already saw that the model development time is very small compared to the run time of beam search without auto-encoding. This benefit will only become more apparent for larger datasets. On top of that, tuning and training the auto-encoder only has to occur once. Thus, in case multiple algorithms must be run, this can all occur with the same encoded features. Similarly, if one has to investigate a similar dataset with new instances every once in a while (e.g. monthly fraud detection investigation), the auto-encoder does not need to be re-trained and the same model can still be used. Hence, in several scenarios, the model development time is still not a deal-breaker if the initial development time is longer than a one-time run of the model.

Another potential limitation of feature reduction using auto-encoding is the decrease in interpretability. When creating patterns from selectors that include the original attributes of the dataset, all rules can directly be read and interpreted. This allows for clear interpretation of the rules and one can find a logic based on these rules. For encoded features, however, you do not know the meaning of the attributes and therefore the developed rules are hard or impossible to interpret. While this inevitable loss of interpretability will always be present, this does not mean that the subgroups become unusable. In some scenarios, like fraud detection, people will mainly be interested in the instances that are in a subgroup. Then, the interpretation of the rules is less relevant. Besides, if you would want to find an intuition behind the subgroups, you can still trace back the instances in the subgroups and inspect their attribute values compared to the general dataset. For example, Fig. 2 displays that the best subgroup found in the Adult dataset has different proportions of education categories compared to the

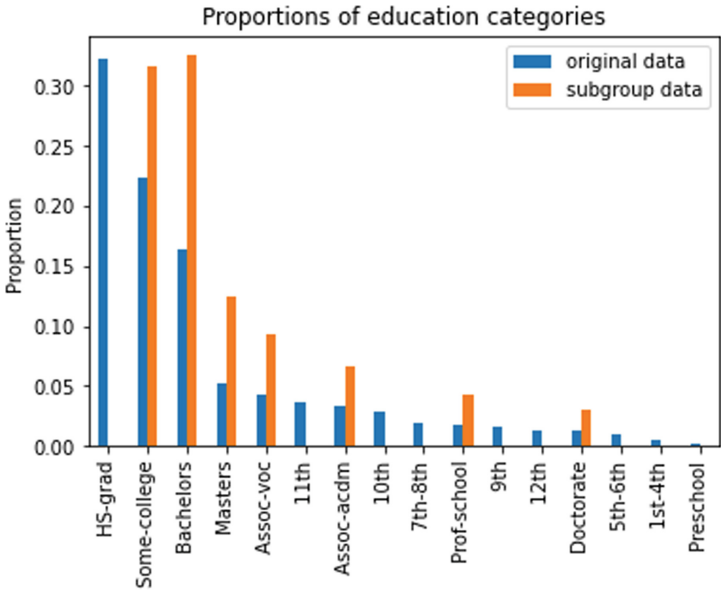


Fig. 2. Adult education category histogram for top subgroup and full dataset.

original dataset. From this, we can clearly see that the subgroup only contains persons with higher education levels. This, intuitively, makes sense when looking for people with a high income. Hence, we conclude that dimensionality reduction in subgroup discovery could still prove to be useful for interpretation. In fact, this expands the subgroup language expressibility: without auto-encoding, a subgroup defined on education level will take the form of an equality constraint to a single value or a set of values; with auto-encoding, a subgroup can be expressed by the skewed orange histogram of education level values.

7 Conclusions

Dimensionality reduction through auto-encoding can improve subgroup discovery (SD) for large datasets. Multiple SD algorithms find subgroups with higher or equivalent quality and better coverage for datasets with a high number of attributes and/or instances when auto-encoding is performed beforehand. On Indoor, the dataset with the largest number of attributes within our experiments, pre-processing through auto-encoding doubles the coverage reached by the BFS, DFS, and APRIORI-SD algorithms (cf. Table 2). With auto-encoding, the beam search algorithm finds subgroups with improved WRAcc quality.

In addition to improved coverage and subgroup quality, the run times of the algorithms are improved greatly. Across Table 2, run times of the algorithms decreased substantially for each algorithm-dataset combination. For datasets with many attributes, we found a decrease in run time of a factor of more than

100. This can have a great impact when handling real-life datasets. For beam search on the Indoor dataset, which has 21 048 instances, the run time decreased from over 13 h to just 3 min while improving coverage and quality. In practice, the number of instances can become even larger leading to an even bigger difference in run times. To achieve this improved run time, the auto-encoder first has to be trained which also takes time. For larger datasets, we found that the decrease in algorithm run time outweighs the model development time.

The increased performance of auto-encoded subgroup discovery comes at the cost of a decrease in interpretability. As the example of Fig. 2 illustrates, intuition can still be derived from the items that are within the subgroup, and the subgroup language becomes more expressive.

In short, we can conclude that using auto-encoding before subgroup discovery is a promising method that can increase the quality, coverage, and run times for subgroup discovery when datasets are large with many attributes.

Future research naturally emerges along two competing axes. On the one hand, we would want to investigate whether we can recover the lost interpretability of subgroups while achieving similar results, by employing interpretability-preserving dimensionality reduction techniques. Straightforward candidates are Principal Component Analysis with constraints on homogeneity and sparsity [4], and the Interpretable Kernel DR algorithm [12]. On the other hand, we would want to investigate whether the lost interpretability allows for better predictive performance, in a LeGo setting [15]: exploiting found subgroups as extra features for multi-label classifiers [7] and as dummy variables in regression models [6, Sect. 8.1] has proven to work; we would want to investigate whether they could be enhanced through auto-encoding.

Acknowledgments. This work is part of the research program Data2People with project EDIC and partly financed by the Dutch Research Council (NWO).

References

1. Abadi, M., Agarwal, A., Barham, P., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). https://www.tensorflow.org/
2. Atzmueller, M.: Subgroup discovery. Wiley Interdisc. Rev. Data Min. Knowl. Discov. **5**(1), 35–49 (2015)
3. Carmona, C.J., González, P., del Jesus, M.J., Herrera, F.: NMEEF-SD: non-dominated multiobjective evolutionary algorithm for extracting fuzzy rules in subgroup discovery. IEEE Trans. Fuzzy Syst. **18**(5), 958–970 (2010)
4. Chipman, H.A., Gu, H.: Interpretable dimension reduction. J. Appl. Stat. **32**(9), 969–987 (2005)
5. Duivesteijn, W., van Dijk, T.C.: Exceptional gestalt mining: combining magic cards to make complex coalitions thrive. In: Proceedings of MLSA (2021)
6. Duivesteijn, W., Feelders, A.J., Knobbe, A.: Exceptional model mining. Data Min. Knowl. Disc. **30**(1), 47–98 (2015). https://doi.org/10.1007/s10618-015-0403-4
7. Duivesteijn, W., Loza Mencía, E., Fürnkranz, J., Knobbe, A.: Multi-label LeGo—enhancing multi-label classifiers with local patterns. In: Hollmén, J., Klawonn, F., Tucker, A. (eds.) IDA 2012. LNCS, vol. 7619, pp. 114–125. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34156-4_12

8. Gamberger, D., Lavrač, N.: Expert-guided subgroup discovery: methodology and application. J. Artif. Intell. Res. **17**, 501–527 (2002)
9. Grosskreutz, H., Rüping, S.: On subgroup discovery in numerical domains. Data Min. Knowl. Disc. **19**(2), 210–226 (2009)
10. Herrera, F., Carmona, C.J., González, P., del Jesus, M.J.: An overview on subgroup discovery: foundations and applications. Knowl. Inf. Syst. **29**(3), 495–525 (2011)
11. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
12. Hosseini, B., Hammer, B.: Interpretable discriminative dimensionality reduction and feature selection on the manifold. In: Brefeld, U., Fromont, E., Hotho, A., Knobbe, A., Maathuis, M., Robardet, C. (eds.) ECML PKDD 2019. LNCS (LNAI), vol. 11906, pp. 310–326. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-46150-8_19
13. Kavšek, B., Lavrač, N.: APRIORI-SD: adapting association rule learning to subgroup discovery. Appl. Artif. Intell. **20**(7), 543–583 (2006)
14. Klösgen, W.: EXPLORA: a multipattern and multistrategy discovery assistant. In: Advances in Knowledge Discovery and Data Mining, pp. 249–271 (1996)
15. Knobbe, A., Crémilleux, B., Fürnkranz, J., Scholz, M.: From local patterns to global models: the LeGo approach to data mining. In: Proceedings of LeGo workshop @ ECMLPKDD, pp. 1–16 (2008)
16. Konijn, R.M., Duivesteijn, W., Kowalczyk, W., Knobbe, A.: Discovering local subgroups, with an application to fraud detection. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) PAKDD 2013. LNCS (LNAI), vol. 7818, pp. 1–12. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37453-1_1
17. Lavrač, N., Flach, P., Zupan, B.: Rule evaluation measures: a unifying view. In: Džeroski, S., Flach, P. (eds.) ILP 1999. LNCS (LNAI), vol. 1634, pp. 174–185. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48751-4_17
18. Lavrač, N., Kavšek, B., Flach, P., Todorovski, L.: Subgroup discovery with CN2-SD. J. Mach. Learn. Res. **5**(2), 153–188 (2004)
19. van Leeuwen, M., Knobbe, A.: Diverse subgroup set discovery. Data Min. Knowl. Discov. **25**, 208–242 (2012)
20. Lemmerich, F., Becker, M.: pysubgroup: easy-to-use subgroup discovery in Python. In: Brefeld, U., et al. (eds.) ECML PKDD 2018. LNCS (LNAI), vol. 11053, pp. 658–662. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-10997-4_46
21. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of ICML Workshop on Deep Learning for Audio, Speech and Language Processing (2013)
22. Meeng, M., Knobbe, A.: For real: a thorough look at numeric attributes in subgroup discovery. Data Min. Knowl. Disc. **35**(1), 158–212 (2020). https://doi.org/10.1007/s10618-020-00703-x
23. Proença, H.M., Klijn, R., Bäck, T., van Leeuwen, M.: Identifying flight delay patterns using diverse subgroup discovery. In: Proceedings of SSCI, pp. 60–67 (2018)
24. Riffenburgh, R.H.: Linear discriminant analysis. Ph.D. thesis, Virginia Polytechnic Institute (1957)
25. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000)
26. Thorndike, R.L.: Who belongs in the family? Psychometrika **18**(4), 267–276 (1953)
27. Wang, Y., Yao, H., Zhao, S.: Auto-encoder based dimensionality reduction. Neurocomputing **184**, 232–242 (2016)
28. Wold, S., Esbensen, K., Geladi, P.: Principal component analysis. Chemom. Intell. Lab. Syst. **2**(1–3), 37–52 (1987)

29. Wrobel, S.: An algorithm for multi-relational discovery of subgroups. In: Komorowski, J., Zytkow, J. (eds.) PKDD 1997. LNCS, vol. 1263, pp. 78–87. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63223-9_108
30. Xu, B., Wang, N., Chen, T., Li, M.: Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853 (2015)
31. Zimmermann, A., De Raedt, L.: Cluster-grouping: from subgroup discovery to clustering. Mach. Learn. **77**(1), 125–159 (2009)