Bitpaths: compressing datasets without decreasing predictive performance

Loren Nuyts[1 [0000−0002−4479−3781]], Laurens Devos[1 [0000−0002−1549−749X]], Wannes Meert[1 [0000−0001−9560−3872]], and Jesse Davis[1 [0000−0002−3748−9263]]

Department of Computer Science, KU Leuven, Celestijnenlaan 200A, Leuven, Belgium
firstname.lastname@kuleuven.be

Abstract. The ever growing amount of data becomes available necessitates more memory to store it. Machine learned models are becoming increasingly sophisticated and efficient in order to navigate this growing amount of data. However, not all data is relevant for a certain machine learning task and storing that irrelevant data is a waste of memory and power. To address this, we propose bitpaths: a novel pattern-based method to compress datasets using a random forest. During inference, a KNN classifier then uses the encoded training examples to make a prediction for the encoded test example. We empirically compare bitpaths' predictive performance with the uncompressed setting. Our method can achieve compression ratios up to 80 for datasets with a large number of features without affecting the predictive performance.

Keywords: feature-encoding · tree-embedding · dataset-compression

1 Introduction

The ever increasing sizes of data poses challenges. On the one hand, more storage is needed. On the other hand, machine learning (ML) approaches runtime scales with size and dimensionality of the data. From a ML perspective, ideally the data could be compressed in a way that still enables good predictive performance [17]. A variety of techniques have been proposed for this task such as product quantization-based approach [11], using a neural network [15], or the pattern-mining based KRIMP [18,12]. A drawback to product quantization is that it is only applicable to real-valued data whereas KRIMP is based on itemsets and hence is only applicable to discrete data.

This paper proposes a pattern-mining-based compression scheme using random forests. Random forests are a popular and powerful method that construct an ensemble of decision trees learned on random subsets of the data. The value predicted for an input example is determined by its *output configuration* [4] – the ordered set of leaves that are activated by the example in each tree – and is obtained by combining the predictions of the individual leaves using e.g. a voting scheme. A decision tree effectively compresses the data by (1) identifying

relevant patterns by automatically selecting predictive features, and (2) grouping together examples that are similar, ignoring differences that are irrelevant to the task at hand. As the number of trees in an ensemble is relatively small, and because each leaf can be represented by a small code (at most d bits, with d the tree's depth), the concatenation of the leaf codes in an example's output configuration is an effective compressed representation of the example. A tree-based scheme has the added benefit that it can naturally cope with data that contains both discrete and real-valued features.

Based on these insights, we developed *bitpaths*, a method that trains a random forest on the original feature space $\mathscr{F}$. The random forest maps each example from the original feature space $\mathscr{F}$ to the encoded output configuration space $\mathscr{B}$. During inference, the encoded output configuration of the test example is computed and a KNN classifier is used on the encoded output configurations of the training examples to make a prediction for the encoded test example.

The method we developed for compressing the dataset is similar to the method Pliakos et al., 2016 [16] used for unsupervised learning tasks: Extremely Random Clustering tree Paths (ERCP). However, they use a different encoding for the output configurations that is not suitable for compression (section 2). Indeed, depending on the dimensions of the random forest, ERCP usually expands the size of the dataset. Bitpaths uses a more memory-efficient encoding while maintaining the same predictive accuracy for supervised learning tasks.

This paper investigates the following 2 key questions to determine whether our proposed method is suitable for compression.

1. How well in terms of accuracy does bitpaths perform when compared to KNN on the original feature space $\mathscr{F}$, RF on the original feature space $\mathscr{F}$ and the related method ERCP [16] (Q1)?
2. How much compression of the training set can be achieved by transforming the original feature space $\mathscr{F}$ to the binary code space $\mathscr{B}$ using *bitpaths* without decreasing predictive performance (Q2)?

2 Preliminaries

Random Forest. A random forest, first proposed by Breiman, 2001 [1], is a randomized decision tree ensemble that is widely used for both classification [2,8] and regression tasks [13,9]. A decision tree ensemble consists of several, independently constructed decision trees. By combining the predictions of all individual trees, the ensemble can overcome the large variance that individual decisions trees usually have [16]. A random forest is such a decision tree ensemble, but it consists of randomized decision trees, which means that each decision tree can only split on a randomly chosen subset of the features.

Output configuration. Given a random forest with m trees, the output configuration (OC) [4] of an example x is the ordered set of leaf nodes $(l_1, .., l_n)$ where each leaf node l_i of the output configuration contains x. The output configuration corresponds to a combination of root-to-leaf paths and the corresponding leaf nodes, where there is one such path and leaf node for each tree in the ensemble.

The output configuration of an example x completely determines the prediction of the random forest for x.

Extremely Random Clustering tree Paths (ERCP) Pliakos et al., 2016 [16] developed a similar method: Extremely Random Clustering tree Paths (ERCP). They use an ensemble of extremely randomized trees instead of a random forest with randomized trees. The most important difference however is the encoding of the output configurations. Instead of encoding the root-to-leaf path, they encode the presence of an example in each node of the tree. Given a tree T with nodes $n_1, .., n_k$, an example $x \in \mathscr{F}$ is encoded as a binary string $b_1..b_k$, where $b_i = 1$ if $x \in n_i$ and 0 otherwise. This results in a very sparse encoding of the example.

3 Bitpaths

The goal of bitpaths is to transform the original feature space $\mathscr{F}$ to the encoded output configuration space $\mathscr{B}$ without losing the essential predictive information. The essential predictive information is extracted by training a random forest on the training set using the original feature space $\mathscr{F}$. By encoding the root-to-leaf paths of each example, the information that the random forest uses to make predictions is kept, while the other information is discarded (section 3.1).

Inference in the encoded output configuration space $\mathscr{B}$ is done with a KNN classifier, which has excellent performance as long as the number of irrelevant features is small. Since our compression scheme removes irrelevant information, KNN is an excellent match.

3.1 Feature construction

First, a random forest of m trees with maximal depth d is trained on the training set using the original feature space $\mathscr{F}$. Second, the path of each training example in each tree is encoded in a binary string. At each node of a tree, the example can take the left branch, in which case a 0 bit is added to the binary code, or the right branch, in which case a 1 bit is added. This results in a binary code of d bits. Figure 1 shows a toy example of a random forest with 3 trees and maximal depth 2. Each leaf node additionally contains the binary code that represents the path from the root to the leaf node. Finally, the encoded OC for the training example is obtained by concatenating the binary codes of each tree in the random forest. In the toy example of figure 1, the training example with $f_1 = f_2 = f_3 = f_4 = 1$ is represented by the encoded OC 01 10 01 and the training example with $f_1 = f_2 = f_3 = f_4 = 5$ is represented by 10 01 00.

3.2 Inference

At inference time, a k-nearest neighbours model predicts the target variable of an encoded test example, based on the encoded OC's of the training examples. The k-nearest neighbours are determined by the Hamming distance between the

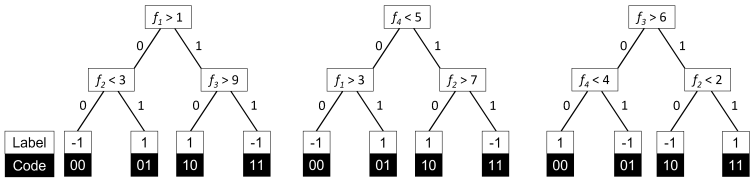

Fig. 1. Example random forest with 3 trees and maximal depth 2. The leaf nodes contain the label and the binary code that represents the root-to-leaf path.

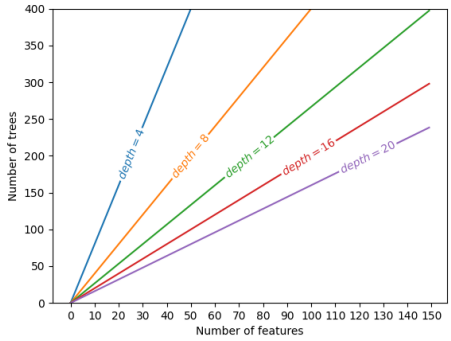

Fig. 2. Lines of equal memory usage in function of the number of trees in the ensemble, the maximal depth of each tree and the number of features in the original feature space.

encoded training examples and the encoded test example. The prediction of the test example is then the average of the predictions of the k-nearest neighbours.

3.3 Compression

For a random forest of m trees and maximal depth d, bitpaths represents each example by an encoded OC of $m * d$ bits. If we assume that each feature in the original feature space $\mathscr{F}$ is represented by a 4 byte float and that there are k features, an example in $\mathscr{F}$ consists of $8 * 4 * k = 32 * k$ bits. Figure 2 shows the lines of equal memory usage in terms of the number of features in $\mathscr{F}$, the number of trees in the random forest and the maximal depth of each tree. If the number of trees and maximal depth is chosen such that you fall below the corresponding depth line, compression is achieved.

The encoding of the related method ERCP on the other hand is not suitable for compression. ERCP represents each example by a binary string of $(2^{d+1} - 1) * m$ bits. This quickly explodes with increasing depth and number of trees and compression would only be possible for datasets with an enormous number of features.

4 Experimental Evaluation

In this section, the following research questions will be answered.

Table 1. Characteristics of the datasets used for evaluation

	Nb of instances	Nb of features
BreastCancer	699	9
Covertype	581 012	54
Higgs	3 468	33
Gina agnostic	601	970
monks-problem-2	250 000	6
ijcnn1	141 691	22
Webspam	350 000	254
tic-tac-toe	958	9
Scene	2 407	299
Fashion MNIST	14 000	784

1. How does bitpaths compare to KNN on the original feature space $\mathscr{F}$, RF on the original feature space $\mathscr{F}$, and the related method ERCP [16] in terms of predictive performance (Q1)?
2. How much compression of the training set can be achieved by transforming the original feature space $\mathscr{F}$ to the binary code space $\mathscr{B}$ using *bitpaths* without decreasing the predictive performance (Q2)?

We used 10 datasets[1] that vary in the number of instances and features (table 1) for our experiments. Min-max normalization is first applied to all datasets. We used the implementation of the *RandomForestClassifier* class of scikit-learn version 1.0 with Gini impurity for all random forests used in the experiments. The exact number of trees and the maximal depth depend on the specific experiment. For the other parameters, the default setting of the *RandomForestClassifier* class is used. The evaluation is done with 10-fold cross-validation.

4.1 Experimental evaluation bitpaths (Q1)

For the first research question, we used a random forest with 50 trees with a maximal depth of 8 and selected the 10 nearest neighbours during inference. Table 2 compares bitpaths with KNN evaluated on the original feature space $\mathscr{F}$, the same random forest as was used for feature construction and ERCP [16] (see section 2). It also contains the average and the standard deviation of the rank per method. Although bitpaths has the best average rank and the lowest standard deviation, both the Friedman test [6,7] and the test developed by Iman and Davenport [10] imply that all compared methods do not significantly differ from each other ($\alpha = 0.05$). Furthermore, following the approach proposed by Demsar, 2006 [3] to compare multiple classifiers in a statistically correct way, the Nemenyi test [14], that performs a pair-wise comparison, and the Bonferroni-Dunn test [5], that additionally corrects for the family-wise error

[1] For the Fashion MNIST dataset, only the examples belonging to class 2 and 4 are used to make the classifier binary. This will be denoted as Fashion MNIST (2, 4)

Table 2. Accuracy results for regular k-nearest neighbours (KNN, $k = 10$), the random forest used for feature construction in the bitpaths method (RF), the method proposed by Pliakos et al., 2016 (ERCP) [16] and our proposed method (bitpaths). For each dataset, the rank of each method is given between brackets. The achieved compression (not in percent) by ERCP and bitpaths on each dataset is also included, where a higher compression ratio means that there is more compression and is thus better. The last column gives the average duration (in seconds) of the compression for bitpaths.

	KNN	**RF**	**ERCP**	**bithpaths**	**Compression ERCP**	**Compression bitpaths**	**Compression time bitpaths (s)**
BreastCancer	0.964 (4)	0.969 (1)	0.968 (2)	0.966 (3)	1.13e-2	7.20e-1	1.15e-1
Covertype	0.971 (1)	0.770 (4)	0.905 (3)	0.908 (2)	6.76e-2	4.32	159
Higgs	0.807 (4)	0.826 (1)	0.820 (3)	0.825 (2)	4.13e-2	2.64	11.7
Gina agnostic	0.826 (4)	0.922 (3)	0.937 (1)	0.935 (2)	1.21	77.6	3.55e-1
monks-problem-2	0.809 (4)	0.960 (1)	0.942 (2.5)	0.942 (2.5)	7.51e-3	4.80e-1	1.44e-1
ijcnn1	0.975 (3)	0.964 (4)	0.979 (2)	0.983 (1)	2.76e-2	1.76	8.10
Webspam	0.982 (3)	0.959 (4)	0.984 (1)	0.983 (2)	3.18e-1	20.3	73.4
tic-tac-toe	0.824 (4)	0.926 (2)	0.918 (3)	0.948 (1)	1.13e-2	7.20e-1	1.34e-1
Scene	0.950 (1)	0.908 (4)	0.926 (3)	0.931 (2)	3.74e-1	23.9	2.59e-1
Fashion MNIST (2, 4)	0.870 (3)	0.868 (4)	0.875 (2)	0.881 (1)	9.82e-1	62.7	1.40
average rank	3.05	2.80	2.25	1.90	-	-	-
standard deviation rank	1.150	1.327	0.750	0.663	-	-	-

in multiple hypothesis testing, conclude that our proposed method doesn't significantly differ from any of the other methods in terms of accuracy. However, 7 of the 10 datasets are compressed with a compression ratio ranging between 1.76 and 77.6 (not in percent), depending on the dataset. This implies that bitpaths can compress datasets without affecting the predictive performance. This stands in contrast with ERCP that uses more memory for 9 of the 10 evaluated datasets. Furthermore, bitpaths compresses datasets quickly: depending on the size of the dataset it takes less than a second or up to a few minutes.

4.2 Compression versus accuracy (Q2)

For the second research question, we varied the compression ratio of the bitpaths algorithm to investigate its effect on the accuracy. The number of trees in each ensemble are chosen such that a compression ratio of 1, 2, 4, 6, 8, 10, 20, 30, 40, 50, 60, 70 and 80 is achieved. The maximal depth of each tree always remained 8. Figure 3 shows that the datasets can be divided in two categories:

1. For the datasets with a low number of features, the best accuracy is found when no compression takes place (compression ratio = 1) and the accuracy gradually decreases with a higher compression ratio. The decrease in accuracy is because the ensembles get smaller and smaller, until eventually they are unable to accurately capture the relationship between the features and target variable. For the breastcancer, monks-problem-2 and tic-tac-toe datasets, a compression ratio beyond 40, 30 and 30 respectively couldn't be

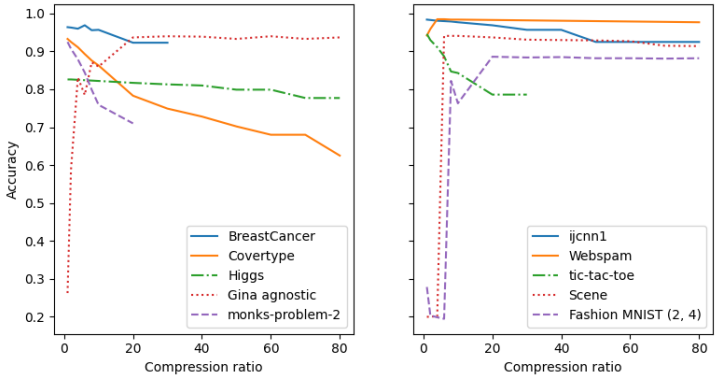

Fig. 3. Evolution of the accuracy of bitpaths in terms of the compression ratio (not in percent) for each example in the training set. We evaluated the following ratios: 1, 2, 4, 6, 8, 10, 20, 30, 40, 50, 60, 70, 80. The number of trees in each ensemble are chosen to achieve such a compression ratio. The maximal depth always remained 8.

achieved because these datasets are a very low dimensionality (< 10) and because we look at depth 8 trees, our codes are always one byte.

2. For the datasets with a high number of features, the initial accuracy is low (except for Webspam) and gradually increases with increasing compression until it reaches its peak. From that point, the accuracy has a steady course and doesn't drop like the datasets with a low number of features. This behaviour can be explained by the extremely large number of trees that are needed when no compression takes place (1000-3000 trees) while this is substantially fewer for the other datasets (maximum 216 trees with no compression). The datasets of this category are too small to properly train such a huge forest, which results in low accuracy results for small compression rates. Webspam on the other hand is large enough to train its large forest, which explains its good initial accuracy. The steady course can also be explained by the number of trees in the ensemble, which for higher compression ratios is still large enough to make good predictions. It is expected that for even higer compression ratio's, the accuracy will also drop.

5 Conclusion

This paper explored how to compress datasets without losing predictive performance. Our approach can handle both real-valued and discrete data by using a random forest to compress the data. The experiments showed that bitpaths can achieve high compression ratios for datasets with many features without affecting the predictive accuracy. One limitation of our approach is that the encoded output configuration is suboptimal in terms of compression when working with non-balanced trees where most examples take the long branches. A different encoding of the output configurations might be more suitable in that case.

Acknowledgements

This work was supported by the Research Foundation-Flanders (1SB1320N to LD), iBOF/21/075, the KU Leuven Research Fund (C14/17/070), and the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" program.

References

1. Breiman, L.: Random forests. Machine Learning **45**, 5–32 (10 2001)
2. Cutler, D.R., Edwards Jr., T.C., Beard, K.H., Cutler, A., Hess, K.T., Gibson, J., Lawler, J.J.: Random forests for classification in ecology. Ecology **88**(11), 2783–2792 (2007)
3. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7**, 1–30 (01 2006)
4. Devos, L., Meert, W., Davis, J.: Adversarial example detection in deployed tree ensembles (2022)
5. Dunn, O.J.: Multiple comparisons among means. Journal of the American Statistical Association **56**(293), 52–64 (1961)
6. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. of the Amer Stat Assoc **32**(200), 675–701 (1937)
7. Friedman, M.: A Comparison of Alternative Tests of Significance for the Problem of m Rankings. The Annals of Mathematical Statistics **11**(1), 86 – 92 (1940)
8. Gislason, P.O., Benediktsson, J.A., Sveinsson, J.R.: Random forests for land cover classification. Pattern Recognition Letters **27**(4), 294–300 (2006), pattern Recognition in Remote Sensing (PRRS 2004)
9. Gong, H., Sun, Y., Shu, X., Huang, B.: Use of random forests regression for predicting iri of asphalt pavements. Construction and Building Materials **189**, 890–897 (2018)
10. Iman, R.L., Davenport, J.M.: Approximations of the critical region of the friedman statistic. Communications in Statistics p. 571–595 (1980)
11. Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(1), 117–128 (2011)
12. Makhalova, T., Kuznetsov, S.O., Napoli, A.: Numerical pattern mining through compression. pp. 112–121 (2019)
13. Montillo, A., Ling, H.: Age regression from faces using random forests. In: 16th IEEE International Conference on Image Processing. pp. 2465–2468 (2009)
14. Nemenyi, P.B.: Distribution-free multiple comparisons. Ph.D. thesis, Princeton University (1963)
15. Park, J., Park, H., Choi, Y.J.: Data compression and prediction using machine learning for industrial iot. In: 2018 International Conference on Information Networking (ICOIN). pp. 818–820 (2018)
16. Pliakos, K., Vens, C.: Feature induction based on extremely randomized tree paths. Online proceedings pp. 3–18 (2016)
17. Sculley, D., Brodley, C.: Compression and machine learning: A new perspective on feature space vectors. In: Data Compression Conference. pp. 332–341 (2006)
18. Vreeken, J., Van Leeuwen, M., Siebes, A.: Krimp: mining itemsets that compress. Data Mining and Knowledge Discovery **23**(1), 169–214