Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Survey Track

A Brief History of Learning Symbolic Higher-Level Representations from Data (And a Curious Look Forward)

Stefan Kramer
Johannes Gutenberg University Mainz, Germany
kramer@informatik.uni-mainz.de

Abstract

Learning higher-level representations from data has been on the agenda of AI research for several decades. In the paper, I will survey various approaches to learning *symbolic higher-level representations*: feature construction and constructive induction, predicate invention, propositionalization, pattern mining, and mining time series patterns. Finally, I will give an outlook on how approaches to learning higher-level representations, symbolic and neural, can benefit from each other to solve current issues in machine learning.

1 Introduction

While the ability to construct complex high-level features from low-level features is widely acknowledged for deep (and also shallow) neural networks, learning *symbolic higher-level representations*[1] has not been much in the focus of attention recently. However, learning symbolic higher-level representations has been studied as well, in various forms, at least from the second half of the 1980s up until today. The reason for both efforts, neural and symbolic, is the belief that learning higher-level representations and in fact, hierarchies of abstractions, is crucial for developing intelligent systems. In this paper, I will focus on symbolic higher-level representations and give an overview and discussion of different approaches along those lines: feature construction and constructive induction (from features to features, see Section 2), predicate invention (from predicates to predicates, see Section 3), propositionalization (from predicates to features, see Section 4), pattern mining (from data to patterns, see Section 5), and time series patterns and shapelets (from time series to time series patterns/shapelets, see Section 6). These approaches are presented side by side in this paper and discussed in one systematic scheme: I will start with the definition of the terms, the purpose, the (expected) benefits, and the promise of that family of methods. Next, I will present a short classification and scheme of the different methods within that family of methods. Third, I will list a few example methods from that category. Fourth, I will discuss open problems (and *problems that may remain open*), a current view, and (potentially) current research in that area. Each section will be concluded by a few pointers to implementations. The list of methods and approaches is by no means comprehensive. Many are left out for brevity and in the interest of discussing all these families of methods in one paper. In Section 7, an outlook on future work on the basis of this part of the literature is given.

[1] We could have used "learning symbolic representations from data" instead of "learning symbolic higher-level representations", but we wanted to stress the goal of ultimately having layered, hierarchical representations, which is currently only achieved for new predicates [Muggleton et al., 2018] and sequences [Wang et al., 2018]. In any case, it is not the intention to discuss higher-order representations in the sense of logics or functional programming.

2 Feature Construction and Constructive Induction

Definition, purpose, benefits, and promise. Methods for *constructive induction* automatically perform a representation change of the input data for a given learning task. In the context of propositional learning, constructive induction is achieved by *feature construction*, the construction of new features from existing features, often by logical connectives or arithmetic operators. The goal is to overcome representational shortcomings, to find more compact and more accurate hypotheses or models.

Classification and scheme. According to [Matheus & Rendell, 1989], methods for constructive induction (i) need to detect whether a change of representation is necessary (*detection*), (ii) need to select new features for inclusion in the new representation (*selection*), (iii) optionally generalize feature definitions (*generalization*), and (iv) potentially have to decide whether to discard features along the way (*evaluation*).

Detection is necessary, because the construction of irrelevant features affects learning similar to noise in the data [Wnek & Michalski, 1992]. Moreover, the complexity of the definitions has to be added to the complexity of the hypotheses. Since the set of features that can be constructed is potentially very large and its detailed evaluation is intractable in general, only a small subset can be included in the representation. In the propositional case, *selection* in constructive induction corresponds to the well-known problem of feature

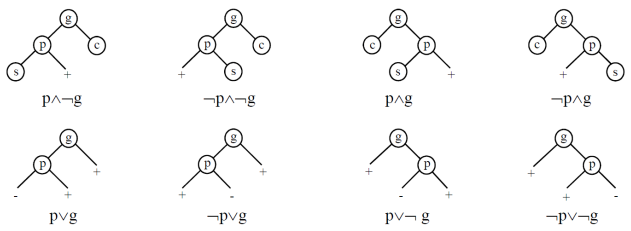

Figure 1: Feature construction in FRINGE [Pagallo & Haussler, 1990] (upper row) and DCFringe [Yang et al., 1991] (lower row) based on recurring patterns near the fringe of a decision tree

selection. *Generalization* may be necessary if a new term is too specific to be useful, but is part of only few approaches. *Evaluation* is necessary if a feature turns out to be less useful than expected in the process downstream.

[Wnek & Michalski, 1992] introduced a taxonomy of constructive induction systems according to the source of information that is used by the constructive induction operators. So the main distinction is different from the one in Matheus' framework, which is based on aspects like detection and selection:

- *Data-driven constructive induction (DCI)*: DCI analyzes the training examples in order to perform constructive induction. Specifically, new descriptors are found by the search for correlations and dependencies among examples, attributes, and concepts.
- *Hypothesis-driven constructive induction (HCI)*: HCI refers to methods that transform representation spaces by analyzing learned hypotheses. Methods for hypothesis-driven constructive induction typically construct new features in iterations, where each iteration involves a learning step and a step which constructs new features based on the hypotheses from the learning step.
- *Knowledge-driven constructive induction (KCI)*: These systems use expert-provided domain knowledge to construct new features. Further, representation changes can be validated by domain experts.
- *Multi-strategy constructive induction (MCI)*: MCI systems combine different approaches to the transformation of feature spaces.

Example methods. Two early and notable example are FRINGE [Pagallo & Haussler, 1990] for the construction of features from decision trees (see Figure 1) and DUCE [Muggleton, 1987] for the construction of features from learned rule sets (see Table 1). Both approaches (amongst others) search for repeated occurrences: FRINGE at the fringe of decision trees, in the original version for conjunctive features (Figure 1, upper row) and in an extended version for disjunctive features (Figure 1, lower row). DUCE applies a number of operators to compress its rule sets, one being the inter-construction operator that turns regularities in the premises of rules into new features (see Table 1(a)).

Problems, current view, and current research. The problem is the evaluation with respect to a ground truth and also comprehensibility. It is easy to construct a lot features, but it is hard to assess how good and useful they are. In neural networks, by constrast, construction and evaluation of feattures are done simultaneously; here they are separate. Another problem is comprehensibility: Newly defined features are most useful if they provide abstractions or generalizations that human users are able to interpret. Ensuring this clearly is not an easy task.

$$B \wedge C \wedge D \wedge E \rightarrow X$$
$$A \wedge B \wedge D \wedge F \rightarrow Y$$
$$\Rightarrow$$
$$B \wedge D \rightarrow Z$$
$$C \wedge E \wedge Z \rightarrow X$$
$$A \wedge F \wedge Z \rightarrow Y$$

$$B \wedge C \wedge D \wedge E \rightarrow X$$
$$A \wedge B \wedge D \wedge F \rightarrow X$$
$$\Rightarrow$$
$$C \wedge E \rightarrow Z$$
$$A \wedge F \rightarrow Z$$
$$B \wedge D \wedge Z \rightarrow X$$

Table 1: Operators in DUCE: (a) *inter-construction* of new feature Z based on patterns in premises of rules (here: $B \wedge D$). (b) *intra-construction* by putting the "residues" of commonalities of clauses into the definition of new feature Z (here: C and E resp. A and F).

In learning of decision trees or rules, other topics are currently pursued, like extremely randomized learning schemes [Geurts et al., 2006] or schemes that frame decision tree learning as an optimization task with respect to a loss function [Chen & Guestrin, 2016]. Complete search in the space of decision trees has also become feasible under syntactic constraints [Nijssen & Fromont, 2010], such that the construction of features (to overcome representational shortcomings concerning the language and algorithmic bias) is not necessary anymore. Simplifying or compressing decision trees or rule sets by intermediate definitions is currently not a popular research topic.

Implementations do not seem to be available at this point of time. However, learning features with discrete definitions is actively investigated in predicate invention and propositionalization (see the next two sections).

3 Predicate Invention

Definition, purpose, benefits, and promise. Learning in relational domains (with objects described by relations and with relational background knowledge) depends on the available vocabulary, i.e., the available predicate, function, and constant symbols. If the vocabulary is limited for a learning task at hand or the task to be learned relies in a complex way on the available base representation, then the invention of new predicates, defined in terms of existing predicates, can contribute to more compact, more accurate, and better comprehensible models (see, e.g., the definition of `stairs` in Table 2(a)) based on visual elements).

```
stair(X, Y)  :- stair1(X, Y).
stair(X, Y)  :- stair1(X, Z), stair(Z,Y).
stair1(X, Y) :- vertical(X, Z),
                horizontal(Z, Y).
```

```
P(X, Y) :- Q(Y, X).
P(X, Y) :- Q(X, Z), R(Z, Y).
```

```
clock_angle(O, H, A) :- highlight(O, H),
     convex(O), light_source(L),
     light_source_angle(O, L, A).

clock_angle(O, H, A) :-
     highlight(O, H), concave(O),
     light_source(L),
     light_source_angle(O, L, A1),
     opposite(A1, A).
```

Table 2: Predicate invention: (a) Predicate `stair1` is defined to be a sequence of a vertical and a horizontal element. (b) Some second-order schemes for learning rules and inventing new predicates in so-called *meta-interpretive learning* (METAGOL). (c) Use of complex new predicates like `convex` in the context of *logical vision*, where occurrences of shadow and light are learned depending on the position of a light source, its angle, and the shape of an object [Muggleton et al., 2018].

Classification and scheme. We can distinguish between *reformulation approaches*, *demand-driven approaches* and *clause-refinement methods* [Stahl, 1993; Kramer, 1995]. *Reformulation approaches* introduce new intermediate predicates as a reformulation of an existing theory in order to express it more compactly. This is done in any case, not only if learning "fails" in the given representation. The knowledge base can be compressed either by *inverse resolution* or by a *schema-driven approach*. The input of compression algorithms are usually clauses, either instances of the training set or disjuncts of a previously learned or partial hypothesis. *Inverse resolution* [Muggleton & Buntine, 1988] is a method for "factoring out" the generalization of two or more clauses, and assigning the "residues" to a new predicate (analogously to the *intra-construction* operator for the definition of new propositional features, see Table 1 (b)). This results in a new predicate describing the variation of the input clauses relatively to their common generalization. *Schema-driven methods* use second-order clauses (see Table 2 (b)) for predicate invention. Newly defined predicates are simply instantiations of predefined or induced schemata. A schema can be instantiated by turning its predicate variables into predicate symbols. In contrast to reformulation approaches, *demand-driven systems* aim to detect situations where the given vocabulary is insufficient for the learning task at hand. *Clause-refinement methods* [Kramer, 1995] make an over-general clause consistent by adding a literal which contains a new predicate. Before the actual refinement step, we have to determine the clause of the theory that is to blame for the incorrectly covered instances. Clause-refinement methods depend on the existence of negative examples, because they aim to discriminate between positive and negative instances. Methods for inverse resolution only take positive instances or clauses of a theory as input. Methods for learning recursive new predicates, e.g., by schemata, characterize positive examples in a special way: They describe everything that the given examples have in common, namely their base-case and the repeated application of an operator.

Example methods. CIGOL [Muggleton & Buntine, 1988] introduced inverse resolution and mostly employed the *intra-construction operator*, whereas RINCON [Wogulis & Langley, 1989] employed the *inter-construction operator* to invent new predicates. In CIGOL, the intra-construction operator is called W-operator, and it is based on (and enabled by) Plotkin's *least general generalization (lgg)* and *relative least general generalization (rlgg)* [Plotkin, 1970; Plotkin, 1971]. Generalised Closed-World Specialisation (GCWS) [Srinivasan et al., 1992] is a way to obtain structured theories in inductive logic programming. It is able to invent predicates for exceptions and exceptions of exceptions, etc., as is needed for learning complex concepts like the one of an intercalary year. GCWS is an example for a clause-refinement method. *Statistical predicate invention* [Kok & Domingos, 2007] iteratively refines clusters of symbols based on the clusters of symbols they appear in atoms with, allowing for multiple cross-cutting clusterings. METAGOL [Muggleton et al., 2018] is a current and very successful schema-driven approach. It relies on the availability of a library of meta-schemata as an inductive bias. Given a suitable library of templates, it is able to learn quite deep sets of clauses, with recursion as well as newly invented predicates.

Problems, current view, and current research. The problems with predicate invention are similar to those with feature construction: too many options to invent new predicates and too few options to assess the utility of candidates. Comprehensibility of new predicates is also a requirement that is hard to meet. Meta-interpretive learning (see METAGOL), although in its central idea not entirely new, has enabled substantial progress recently. One may argue that the availability of a suitable template library is a limiting factor, however, hyperparameter optimization and the search for suitable network architectures in current deep learning does not appear much easier in comparison.

Implementations. In contrast to feature construction, implementations are available, e.g., GCWS is available in Aleph[2], and METAGOL[3] can be downloaded from github.

4 Propositionalization

Definition, purpose, benefits, and promise. Propositionalization is the representation change of transforming a rela-

[2] https://www.cs.ox.ac.uk/activities/programinduction/Aleph

[3] https://github.com/metagol/metagol

```
?- person(K), parent(K, Y), has_pet(Y, cat).
```

```
p(K, Z) :- person(K), has_account(K, Y),
           overdraft(Y, Z).
```

Table 3: Propositionalization: (a) A conjunctive query with key `K` defining a Boolean feature for an instance `K` (`true` if it succeeds for some `K`, `false` otherwise). This is called an *existential feature*. (b) A clause defining a feature for a relational example. `K` denotes the key (i.e., the identifier of the instance). It is assumed that a person may have several accounts `Y`, each with a different numerical overdraft limit `Z`. Such a clause may be the basis for several types of features, e.g., by applying aggregate functions on the set of different values for `Z`. This is called an *aggregate feature*.

tional representation of a learning problem into a propositional (feature-based, attribute-value) representation [Kramer et al., 2001]. The idea is to build higher-level feature representations from lower-level relational data, just like super-pixels are constructed from pixels in image data. Taking such an approach, feature construction can be decoupled from model construction. In propositionalization, the search space of relational features is not gradually searched and expanded as in inductive logic programming (ILP), but all relational features with certain properties, e.g., up to some maximal syntactic size or within a minimal and maximal frequency of occurrence, are generated and used to transform the representation. The advantage is that it is possible to take advantage of any progress with propositional learning algorithms in this way. The disadvantage is the potential loss of information due to size or frequency constraints. If propositionalization does not give the desired results, it should at least be used as a baseline to show that more advanced search and optimization strategies are worth the effort.

Classification and scheme. Propositionalization schemes can be categorized according to the types of features that are constructed. One basic distinction is the one between *existential features* and *aggregate features* (see Table 3). *Existential features* are defined by *conjunctive queries*, which, when succeeding for an instance, give the value `true`, and `false`, otherwise. Aggregate features [Krogel & Wrobel, 2001] are more complex: We consider a defined set of variables except the key, which give the answer substitutions for that query (when the key variable is bound to some instance identifier). In the above example (see Table 3 (b)), `K` is the key, and the user has defined variable `Z` to be the one of interest for relational feature construction. When the query is evaluated for some instance `K`, we gather all values for `Z`, and, in the final step, apply a user-defined set of aggregate functions (like minimum, maximum, mean, standard deviation, mode, etc.) to that set of values to define propositional values. Clearly, variants are possible: The clause in Table 3 (b) can be the basis for some test against a threshold (e.g., `Z > 10000`), or it can be used not to turn the problem into a propositional problem, but, without aggregates, into a *multi-instance learning* or *multi-tuple learning problem* [De Raedt, 2008].

Example methods. The first methods were due to Nada Lavrač and Sašo Džeroski [Lavrac et al., 1991], methods based on clauses, either syntactically constrained and evaluated by some score function [Flach & Lachiche, 2001] or constrained by minimum frequency (minimum support) [Dehaspe & Toivonen, 1999] were following. WARMR [Dehaspe & Toivonen, 1999], the frequent pattern mining variant for conjunctive queries, can also be regarded as the first approach to graph mining, although it was proposed in an even more expressive formalism. Propositionalization based on aggregate functions was proposed only slightly later [Krogel & Wrobel, 2001].

Problems, current view, and current research. Many of the transformation methods still belong to the standard repertoire for preprocessing relational datasets today. Two recent approaches are *wordification* and *dynamic* or *lazy propositionalization*. *Wordification* [Perovsek et al., 2015] considers an instance (i.e., a row in a table) as a document and individual attribute values as words in this document. More complex words can be formed from building *n*-grams from simple words, and taking into account foreign key relations among tables. Documents are transformed into a bag-of-words representation by calculating TF-IDF values for each word of each document. *Dynamic propositionalization* [Schouterden et al., 2019] constructs these features in a lazy manner that is guided by the learner. The gradual expansion of the feature table resembles the way how ILP systems gradually construct longer clauses by first constructing shorter ones and considering only the promising ones for further expansion. Answer substitutions are cached for further iterations. Overall, lazy propositionalization blurs the line between propositionalization (a static transformation scheme, as defined above) and relational learning (dynamic search through the space of clauses/features), but still may be a promising middle ground with advantages neither of the two extremes have: less loss of information than traditional propositionalization and faster running times than traditional relational learning.

Implementations. Functionality for the construction of relational features based on learned clauses can again be found in Aleph (see above). Tertius [Flach & Lachiche, 2001], RELAGGS [Krogel & Wrobel, 2001], and wordification [Perovsek et al., 2015] are implemented in the relational data mining module (https://github.com/xflows/rdm) of the web-based data mining platform ClowdFlows (available at http://clowdflows.org).

5 Pattern Mining

Definition, purpose, benefits, and promise. Pattern mining is concerned with finding patterns of interest in a given database. Patterns are defined to be elements of so-called *pat-*

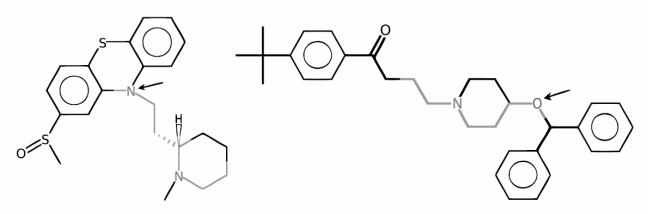

Figure 2: Graph mining results in two example molecules [Maunz et al., 2010]. The identified pattern is shown in gray and occurs in both examples. Note that the pattern occurrences are not identical (N representing nitrogen is at one end of the pattern occurrence in the left molecular graph, and O representing oxygen is at one end of the right molecular graph). Regarding electronegativity, both N and O are treated equally in the graph matching procedure.

tern languages. An *interestingness predicate* picks the subset of patterns from that language that is interesting for a human user. The predicate can be defined as a logical expression combining criteria (constraints) that need to be fulfilled for a pattern to count as interesting. This definition works well for a variety of patterns in discrete data such as sets, multi-sets, strings, sequences, trees, graphs, hypergraphs, and logical representations. Generalization to the continuous domain is not straightforward, and not many convincing proposals exist. In the simplest case, the pattern language L is the powerset $P(I)$ of a set of possible items I, in which case we speak of itemsets. Figure 2 shows the occurence of a subgraph pattern (in gray) in two molecules from a real-world application. Another example, from Table 3(a), shows a relational pattern that is represented by a conjunctive query. Relational patterns subsume other pattern domains, in other words, everything from itemsets via graphs to hypergraphs can be represented in that way, however, clearly more efficient algorithms exist for "native" representations. Overall, the beauty of the approach is that a complete list of solutions is generated, not any arbitrary list from greedy search, meta-heuristics, or local optima from gradient descent or ascent.

The first use of pattern mining is in exploratory data analysis, specifically for the construction of association rules. This works essentially for any pattern language, as long as a generality order on the patterns is defined. A generality order $\leq$ defines a pattern X to be more general (or equal) than Y, written as $X \leq Y$, if it holds that whenever pattern Y occurs in an instance, then also X occurs in it (in all possible worlds, i.e., in all conceivable databases). One can imagine pattern X to be "contained" in pattern Y. Then it is easy to construct association rules of the form $X \rightarrow Y$ (that hold with a certain probability; for the other way round, $Y \rightarrow X$, the probability is 1.0). That estimated probability of $P(Y|X)$ is usually called *confidence*. Such (unsupervised) rules can be browsed by analysists and used for data exploration. However, the one use of patterns that is *much more common in practice* is to use patterns in variables in other models, e.g., in classification, regression, clustering, or generative models. In this sense, patterns can be very useful for building higher-level symbolic representation from low-level data. Here, the raw, low-level data are, e.g., itemset data, sequences, or graphs, and the higher-level representations include patterns in those data, which enable a more abstract view.

Classification and scheme. The classification of method depends on *pattern languages* (also called *pattern domains*), *constraints*, and *solution strategies* and *search strategies* of the solvers. For simple minimum and maximum frequency constraints, classical level-wise search [Mannila & Toivonen, 1997] can take advantage of the monotonicity or anti-monotonicity of frequency-related constraints and thus effectively prune the search space. For class-correlated pattern mining with a convex score function, branch-and-bound strategies can be used [Morishita & Sese, 2000]. Pattern growth algorithms [Han et al., 2017] build data structures from the data, simplify the data along the way, and "read off the patterns" incrementally from that data structure.

Example methods. Classical methods include the pioneering Apriori algorithm [Agrawal & Srikant, 1994] and FP-Growth for itemsets [Han et al., 2017], gSpan for enumerating frequent subgraph patterns [Han et al., 2017], class-correlated pattern mining [Morishita & Sese, 2000], optimal pattern mining algorithms for strings [Fischer et al., 2006], and the tree mining approach FreeTreeMiner [Chi et al., 2003].

Problems, current view, and current research. Although progress has been made in reducing the very large solution sets that often are larger than the original data, the problem with a suitable selection remains. Well-known and tested strategies remove the redundancy in the patterns with regard to their occurrences in the data (keeping, e.g., only so-called *frequent closed* [Pasquier et al., 1999] or *frequent free patterns* [Boulicaut et al., 2003]). Recent years have seen methods for *pattern set mining* [Guns et al., 2013] and *pattern sampling* [Dzyuba et al., 2017]. Pattern set mining focuses on returning a suitable *solution set* in the first place, and not enumerating all solutions and only then considering constraints like pattern diversity and pattern coverage. Pattern set mining is strongly related to rule learning and subgroup discovery. Pattern sampling is the idea of sampling patterns with a probability proportional to their quality (measured by some score function). Pattern diversity and pattern coverage is not automatically taken care of by pattern sampling.

Implementations. Itemset mining is implemented in virtually every machine learning and data mining workbench and library. The only optimal pattern mining algorithms known today are for string mining [Fischer et al., 2006], as the order in strings gives valuable information compared to unordered elements in sets. One of the optimal string mining algorithms can be found on github[4]. Tree mining algorithms

[4] https://github.com/atifraza/MiSTiCl/blob/master/string_miner

for the batch case[5] and the streaming case[6] are also available. Implementations of the classical graph mining algorithm gSpan are available stand-alone[7] or as part of libraries.

6 Time Series Patterns and Shapelets

Definition, purpose, benefits, and promise. Shapelets [Ye & Keogh, 2011] are discriminative time series subsequences that allow generation of interpretable classification models, which provide faster and generally better classification than the nearest neighbor approach, using, e.g, *dynamic time warping* [Ding et al., 2008] as a distance measure. However, the shapelet discovery process requires the evaluation of all possible subsequences of all time series in the training set, making it extremely computationally intensive. Consequently, shapelet discovery for large time series datasets quickly becomes infeasible. A number of improvements have been proposed to reduce the training time. These techniques use approximations, algorithmic improvements, like pruning techniques making use of the triangular inequality or caching, or discretization, and often lead to reduced classification accuracy compared to the exact method.

Classification and scheme. The first distinction is whether we consider the time series completely (in dynamic time warping) or whether we focus on only relatively short subsequences. The former approaches are typically based on variants of *dynamic time warping (DTW)* [Ding et al., 2008], the latter on variants of *shapelets* [Ye & Keogh, 2011; Hills et al., 2014], which are prototypical patterns of shorter length that can occur anywhere in the time series and are usually discriminative[8]. Further distinctions can be drawn: Methods can be categorized according to whether they are complete or whether they are heuristic or based on sampling, and whether they are directly embedded into a specific learning algorithm or whether they can be used in any subsequent algorithm.

Example methods. The original proposal, YK-Shapelets [Ye & Keogh, 2011], includes early candidate pruning using an upper-bound on the information gain and reported a speed-up of three orders of magnitude compared to the brute force approach. Fast-Shapelets [Rakthanmanon & Keogh, 2013] reduces the dimensionality of the data using SAX [Lin et al., 2007] and then performs a random projection based shapelet discovery using this lower dimensional data. It uses a heuristic approach and provides a huge reduction in computational costs, but requires extensive hyperparameter tuning. A recently published approach called Generalized Random Shapelet Forests (gRSF) [Karlsson et al., 2016] employs ensembles and a randomized candidate sampling-based shapelet discovery process for improved classification accuracy and reduced runtime.

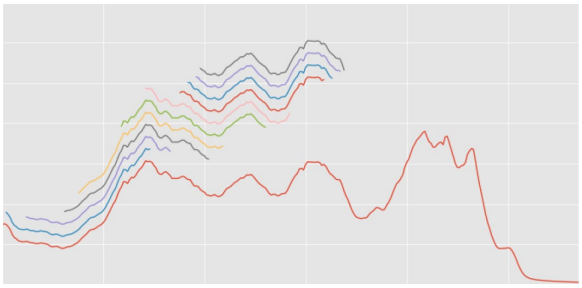

Figure 3: Example shapelets capturing typical time courses for a certain class of time series [Raza & Kramer, 2017]. The algorithm producing this plot, Random-Shapelets, samples shapelets starting at different time points and thus covers different parts of the sequence.

Problems, current view, and current research. Having both fast and exact methods for very large datasets of long time series is still not completely solved. The focus of whole time series and short important subsequences are just two extremes of a spectrum. Any option in between and also complementing each other would be interesting to see, with automatic choice of what is relevant for a given domain and dataset. The same goes for considering *both* the time domain and the frequency domain [Schäfer, 2015; Schäfer, 2016] of time series data. Making using of *optimal string mining* algorithms (see above) for time series analysis is currently being investigated [Raza & Kramer, 2020]. The *matrix profile* [Yeh et al., 2016] (essentially a matrix with the smallest distance to any subsequence for any given subsequence) has been shown to be a great tool for the discovery of sequence patterns.

Implementations. Numerous implementations exist of shapelets. The matrix profile implementation called STUMPY can be found on github[9].

7 Future Directions and Conclusions

Having presented all these approaches in one paper, it is clear that they are not isolated attempts, but tightly related. In that sense, we hope that one field can benefit from a related field, if some advances are made there. Considering the progress with embeddings of different sorts, like word vectors (word2vec, GloVe, etc.) or other vector representations of structured objects (graph2vec, mol2vec, patient2vec, etc.), it is evident that these approaches have an advantage, because they usually can be further optimized and refined, while discrete symbolic structures can only be generated and evaluated in a next step. In other words, these structures do not evolve gradually during some phase of learning. Symbolic chunks of knowledge ideally stand out statistically *and* make sense to humans. Most current meth-

[5] https://github.com/yunchi/FreeTreeMiner
[6] https://github.com/abifet/adatreenat
[7] https://github.com/betterenvi/gSpan
[8] Historically, most of the work on time series in the data mining literature after the year 2000 has focused on classification, although clustering has attracted more and more attention recently.
[9] https://github.com/TDAmeritrade/stumpy

ods only create a single layer of a symbolic representation, not a hierarchy with different levels of abstraction. Recent exceptions are methods for predicate invention [Muggleton et al., 2018] and sequence mining [Wang et al., 2018]. The whole area of *relating continuous types of higher-level representations to symbolic higher-level representations* is, to my knowledge, largely unexplored.

However, deep neural networks might benefit from higher-level symbolic representations like patterns as well. Patterns, for instance, for sequences and time series, can be computed very efficiently and could give neural networks for such data guidance or a prior that may lead to faster training times. Learning from fewer training instances, by unsupervised pre-training, to create a structure that is later "filled" by evidence, is currently a hot research topic (see the recent AAAI 2020 panel of Y. Bengio, G. Hinton, and Y. LeCun). Patterns can also be way to pre-filter instances for training for DNNs [Ahmadi et al., 2018] or to provide some raw material (i.e., building blocks) that can be further refined statistically [Lethan et al., 2013]. Finally, patterns and any kind of symbolic representations are useful to explain the behavior of deep neural networks, as soon such models have to interact with human users. These explanations can be given on a global or a local level. Therefore, we believe that both neural and symbolic higher-level representations and their interrelationship have a role to play in the future of intelligent systems.

References

[Agrawal & Srikant, 1994] R. Agrawal, R. Srikant: Fast Algorithms for Mining Association Rules in Large Databases, in: *Proc. of the 20th International Conference on Very Large Data Bases (VLDB 1994),* 487-499, 1994.

[Ahmadi et al., 2018] Z. Ahmadi, P. Martens, C. Koch, T. Gottron, S. Kramer: Towards Bankruptcy Prediction: Deep Sentiment Mining to Detect Financial Distress from Business Management Reports, in: *Proc. of the 5th IEEE International Conference on Data Science and Advanced Analytics (DSAA 2018)*, 293-302, 2018.

[Boulicaut et al., 2003] J.-F. Boulicaut, A. Bykowski, C. Rigotti: Free-Sets: A Condensed Representation of Boolean Data for the Approximation of Frequency Queries, *Data Mining and Knowledge Discovery*, 7, 5–22, 2003.

[Chen & Guestrin, 2016] T. Chen, C. Guestrin: XGBoost: A Scalable Tree Boosting System, in: *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794, 2016.

[Chi et al., 2003] Y. Chi, Y. Yang, R. Muntz: Indexing and Mining Free Trees, in: *Proc. of the Third IEEE International Conference on Data Mining (ICDM 2003)*, 509, 2003.

[Dehaspe & Toivonen, 1999] L. Dehaspe, H. Toivonen: Discovery of Frequent DATALOG Patterns, *Data Mining and Knowledge Discovery*, 3(1):7-36, 1999.

[De Raedt, 2008] L. De Raedt: *Logical and Relational Learning*, Springer, 2008.

[Ding et al., 2008] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, E. Keogh: Querying and mining of time series data, in: *Proc. of the VLDB Endowment*, 1(2):1542-1552, 2008.

[Dzyuba et al., 2017] V. Dzyuba, M. van Leeuwen, L. De Raedt: Flexible Constrained Sampling with Guarantees for Pattern Mining, *Data Mining and Knowledge Discovery*, 31(5):1266-1293, 2017.

[Fischer et al., 2006] J. Fischer, V. Heun, S. Kramer: Optimal String Mining Under Frequency Constraints, in: *Proc. of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2006)*, 139-150, 2006.

[Flach & Lachiche, 2001] P.A. Flach, N. Lachiche: Confirmation-Guided Discovery of First-Order Rules with Tertius, *Machine Learning*, 42: 61–95, 2001.

[Geurts et al., 2006] P. Geurts, D. Ernst, L. Wehenkel: Extremely randomized trees, *Machine Learning*, 63(1):3-42, 2006.

[Guns et al., 2013] T. Guns, S. Nijssen, L. De Raedt: k-Pattern Set Mining under Constraints, *IEEE Transactions on Knowledge and Data Engineering*, 25(2):402-418, 2013.

[Han et al., 2017] J. Han, M. Kamber, J. Pei: *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2017.

[Hills et al., 2014] J. Hills, J. Lines, E. Baranauskas, J. Mapp, A. Bagnall: Classification of time series by shapelet transformation, *Data Mining and Knowledge Discovery*, 28(4):851-881, 2014.

[Karlsson et al., 2016] I. Karlsson, P. Papapetrou, H. Boström: Generalized random shapelet forests, *Data Mining and Knowledge Discovery*, 30:1053–1085, 2016.

[Kok & Domingos, 2007] S. Kok, P.M. Domingos: Statistical predicate invention, in: *Proc. of the Twenty-Fourth International Conference on Machine Learning (ICML 2007)*, 433-440, 2007.

[Kramer, 1995] S. Kramer: *Predicate Invention: A Comprehensive View*, Technical Report, OFAI-TR-95-32, Austrian Research Institute for Artificial Intelligence, 1995.

[Kramer et al., 2001] S. Kramer, N. Lavrac, P. Flach: Propositionalization Approaches to Relational Data Mining, in:

Dzeroski S., Lavrac N. (Eds.): *Relational Data Mining*, Springer Verlag, Berlin Heidelberg New York, 2001.

[Krogel & Wrobel, 2001] M.-A. Krogel, S. Wrobel: Transformation-Based Learning Using Multirelational Aggregation, in: *Proc. of the 11th International Conference on Inductive Logic Programming (ILP 2001)*, 142-155, 2001.

[Lavrac et al., 1991] N. Lavrac, S. Dzeroski, M. Grobelnik: Learning Nonrecursive Definitions of Relations with LINUS, in: *Proc. of the European Working Session on Learning (EWSL 1991)*, 265-281, 1991.

[Lethan et al., 2013] B. Letham, C. Rudin, T.H. McCormick, D. Madigan: An Interpretable Stroke Prediction Model using Rules and Bayesian Analysis, in: *Proc. of the 17th AAAI Conference on Late-Breaking Developments in the Field of Artificial* Intelligence*,* 2013.

[Lin et al., 2007] J. Lin, E. Keogh, L. Wei, S. Lonardi: Experiencing SAX: a novel symbolic representation of time series, *Data Mining and Knowledge Discover*y, 15(2):107-144, 2007.

[Mannila & Toivonen, 1997] H. Mannila, H. Toivonen: Levelwise Search and Borders of Theories in Knowledge Discovery, *Data Mining and Knowledge Discovery*, 1(3): 241-258, 1997.

[Matheus & Rendell, 1989] C.J. Matheus, L. Rendell: Constructive Induction on Decision Tree, in: *Proc. of the 11th International Joint Conference on Artificial Intelligence (IJCAI 1989)*, 1989.

[Maunz et al., 2010] A. Maunz, C. Helma, T. Cramer, S. Kramer: Latent Structure Pattern Mining, in: *Proc. of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-2010)*, 2010.

[Morishita & Sese, 2000] S. Morishita, J. Sese: Traversing Itemset Lattice with Statistical Metric Pruning, in: *Proc. of the 19th Symposium on Principles of Database Systems (PODS 2000)*, 226-236, 2000.

[Muggleton, 1987] S. Muggleton: DUCE: an Oracle-Based Approach to Constructive Induction, in: *Proc. of the 10th International Joint Conference on Artificial Intelligence (IJCAI 1987)*, 1987.

[Muggleton & Buntine, 1988] S. Muggleton, W. Buntine: Machine Invention of First-Order Predicates by Inverting Resolution, in: *Proc. of the Fifth International Conference on Machine Learning (ICML 1988)*, Morgan Kaufmann, San Mateo, CA, 339-352, 1988.

[Muggleton et al., 2018] S. Muggleton, W.-Z. Dai, C. Sammut, A. Tamaddoni-Nezhad, J. Wen, Z.-H. Zhou: Meta-Interpretive Learning from Noisy Images, *Machine Learning*, 107:1097–1118, 2018.

[Nijssen & Fromont, 2010] S. Nijssen, E. Fromont: Optimal constraint-based decision tree induction from itemset lattices, *Data Mining and Knowledge Discovery*, 21(1):9-51, 2010.

[Pagallo & Haussler, 1990] G. Pagallo, D. Haussler: Boolean Feature Discovery in Empirical Learning, *Machine Learning*, 5:71-97, 1990.

[Pasquier et al., 1999] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal: Discovering Frequent Closed Itemsets for Association Rules, in: *Proc. of the International Conference on Database Theory (ICDT 1999)*, 398-416, 1999.

[Perovsek et al., 2015] M. Perovsek, A. Vavpetic, J. Kranjc, B. Cestnik, N. Lavrac: Wordification: Propositionalization by Unfolding Relational Data into Bags of Words, *Expert Systems with Application,* 42(17-18):6442-6456, 2015.

[Plotkin, 1970] G. Plotkin: A Note on Inductive Generalization, in: Meltzer B., Michie D. (Eds.): *Machine Intelligence*, 5: 153–163, 1970.

[Plotkin, 1971] G. Plotkin: A Further Note on Inductive Generalization, in: Meltzer B., Michie D. (Eds.): *Machine Intelligence*, 6: 101–124, 1971.

[Rakthanmanon & Keogh, 2013] T. Rakthanmanon, E. Keogh: Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets, in: *Proc, of the 2013 SIAM International Conference on Data Mining*, 668-676, 2013.

[Raza & Kramer, 2017] A. Raza, S. Kramer: *Ensembles of Randomized Time Series Shapelets Provide Improved Accuracy while Reducing Computational Costs*. CoRR abs/1702.06712, 2017.

[Raza & Kramer, 2020] A. Raza, S. Kramer: Accelerating Pattern-Based Time Series Classification: A Linear Time and Space String Mining Approach, *Knowledge and Information Systems*, 3, 2020.

[Schäfer, 2015] P. Schäfer: The BOSS is concerned with time series classification in the presence of noise, *Data Mining and Knowledge Discovery*, 29(6):1505-1530, 2015.

[Schäfer, 2016] P. Schäfer: Scalable time series classification, *Data Mining and Knowledge Discovery*, 30(5):1273-1298, 2016.

[Schouterden et al., 2019] J. Schouterden, J. Davis, H. Blockeel: *LazyBum: Decision tree learning using lazy propositionalization*. CoRR abs/1909.05044, 2019.

[Srinivasan et al., 1992] A. Srinivasan, S.H. Muggleton, M. Bain: Distinguishing Noise from Exceptions in Non-Monotonic Learning, in: *Proc. of the Second International*

Workshop on Inductive Logic Programming (ILP 1992), 1992.

[Stahl, 1993] I. Stahl: Predicate Invention in ILP: an Overview, in: *Proc. of the European Conference on Machine learning (ECML 1993)*, 313-322, 1993.

[Wang et al., 2018] K. Wang, E. Sadredini, K. Skadron: Hierarchical Pattern Mining with the Automata Processor, *International Journal of Parallel Programming*, 46:376–411, 2018.

[Wnek & Michalski, 1992] J. Wnek, R.S. Michalski: *Hypothesis-driven Constructive Induction in AQ17-HCI: A Method and Experiments*, Technical Report, Center for Artificial Intelligence, George Mason University, 1992.

[Wogulis & Langley 1989] J. Wogulis, P. Langley: Improving Efficiency by Learning Intermediate Concepts, in: *Proc. of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI 1989)*, Morgan Kaufmann, Los Altos, CA, 657-662, 1989.

[Ye & Keogh, 2011] L. Ye, E. Keogh: Time series shapelets: a novel technique that allows accurate, interpretable and fast classification, *Data Mining and Knowledge Discovery*, 22(1):149-182, 2011.

[Yang et al., 1991] D.-S. Yang. G. Blix, L.A. Rendell: The replication problem: A constructive induction approach, in: *Proc. of the European Working Session on Learning (EWSL 1991)*, 44-61, 1991.

[Yeh et al., 2016] C.C.M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H.A. Dau, D.F. Silva, A. Mueen, E. Keogh: Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View that Includes Motifs, Discords and Shapelets, in: *Proc. of the IEEE International Conference on Data Mining (ICDM 2016)*, 2016.